

10/11/00

JCS23 U.S. PTO

+

# UTILITY PATENT APPLICATION TRANSMITTAL

(Only for new nonprovisional applications under 37 CFR 1.53(b))

Attorney Docket No. 169.1867

First Named Inventor or Application Identifier

ANDREW PETER BRADLEY

Express Mail Label No.

 00/T/00  
 10/11/00  
 JCS23 U.S. PTO  
 169.1867/36

## APPLICATION ELEMENTS

See MPEP chapter 600 concerning utility patent application contents.

1. ☐ Fee Transmittal Form  
(Submit an original, and a duplicate for fee processing)
2. ☐ Applicant claims small entity status.  
See 37 CFR 1.27.
3. ☒ Specification Total Pages
4. ☒ Drawing(s) (35 USC 113) Total Sheets
5. ☐ Oath or Declaration Total Pages
- a. ☐ Newly executed (original or copy)
- b. ☐ Copy from a prior application (37 CFR 1.63(d))  
(for continuation/divisional with Box 17 completed)  
[Note Box 6 below]
- i. ☐ **DELETION OF INVENTOR(S)**  
Signed Statement attached deleting  
inventor(s) named in the prior application, see  
37 CFR 1.63(d)(2) and 1.33(b).
6. ☒ Application Data Sheet. See 37 CFR 1.76

## ADDRESS TO:

 Commissioner for Patents  
 Box Patent Application  
 Washington, DC 20231

7. ☐ CD-ROM or CD-R in duplicate, large table or Computer Program (Appendix)
8. ☐ Nucleotide and/or Amino Acid Sequence Submission  
(if applicable, all necessary)
- a. ☐ Computer Readable Form (CRF)
- b. Specification Sequence Listing on:
- i. ☐ CD-ROM or CD-R (2 copies); or
- ii. ☐ paper
- c. ☐ Statements verifying identity of above copies
- ACCOMPANYING APPLICATION PARTS**
9. ☐ Assignment Papers (cover sheet & document(s))
10. ☐ 37 CFR 3.73(b) Statement ☐ Power of Attorney  
(when there is an assignee)
11. ☐ English Translation Document (if applicable)
12. ☐ Information Disclosure Statement (IDS)/PTO-1449 ☐ Copies of IDS Citations
13. ☐ Preliminary Amendment
14. ☒ Return Receipt Postcard (MPEP 503)  
(Should be specifically itemized)
15. ☐ Certified Copy of Priority Document(s)  
(if foreign priority is claimed)
16. ☐ Other: \_\_\_\_\_

17. If a CONTINUING APPLICATION, check appropriate box and supply the requisite information.

☐ Continuation ☐ Divisional ☐ Continuation-in-part (CIP) of prior application No. \_\_\_\_/\_\_\_\_\_  
 Prior application information: Examiner \_\_\_\_\_ Group/Art Unit: \_\_\_\_\_

For CONTINUATION OR DIVISIONAL APPS only. The entire disclosure of the prior application, from which an oath or declaration is supplied under Box 5U, is considered a part of the disclosure of the accompanying continuation or divisional application and is hereby incorporated by reference. The incorporation can only be relied upon when a portion has been inadvertently omitted from the submitted application parts.

## 18. CORRESPONDENCE ADDRESS

☒ Customer Number or Bar Code Label ☐ 05514  
 (Insert Customer No. or Attach bar code label here) or ☐ Correspondence address below

NAME				
Address				
City	State	Zip Code		
Country	Telephone	Fax		

CLAIMS	(1) FOR	(2) NUMBER FILED	(3) NUMBER EXTRA	(4) RATE	(5) CALCULATIONS
	TOTAL CLAIMS (37 CFR 1.16(c))	57-20 =	37	X \$ 18.00 =	\$666.00
	INDEPENDENT CLAIMS (37 CFR 1.16(b))	6-3 =	3	X \$ 80.00 =	\$240.00
	MULTIPLE DEPENDENT CLAIMS (if applicable) (37 CFR 1.16(d))			\$270.00 =	\$0
				BASIC FEE (37 CFR 1.16(a))	\$710.00
				Total of above Calculations =	\$1616.00
				Reduction by 50% for filing by small entity (Note 37 CFR 1.9, 1.27, 1.28).	0
				TOTAL =	\$1616.00

## 19. Small entity status


- a. ☐ A small entity statement is enclosed
- b. ☐ A small entity statement was filed in the prior nonprovisional application and such status is still proper and desired.
- c. ☐ Is no longer claimed.

20. ☒ A check in the amount of \$ 1616.00 to cover the filing fee is enclosed.21. ☐ A check in the amount of \$ \_\_\_\_\_ to cover the recordal fee is enclosed.

## 22. The Commissioner is hereby authorized to credit overpayments or charge the following fees to Deposit Account No. 06-1205:

- a. ☒ Fees required under 37 CFR 1.16.
- b. ☒ Fees required under 37 CFR 1.17.
- c. ☐ Fees required under 37 CFR 1.18.

## SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT REQUIRED

NAME	LEONARD P. DIANA	
SIGNATURE		29,296
DATE	October 10, 2000	

#### INVENTOR INFORMATION

Inventor One Given Name:: ANDREW PETER  
Family Name:: BRADLEY  
Postal Address Line One:: 68 The Bulwark, Castlecrag  
City:: New South Wales 2068  
Country:: Australia  
Inventor Two Given Name:: DOMINIC  
Family Name:: YIP  
Postal Address Line One:: 127 Bent Street, Lindfield  
City:: New South Wales 2070  
Country:: Australia  
Inventor Three Given Name:: ANDREW JAMES  
Family Name:: DORRELL  
Postal Address Line One:: 38 Rusden Road, East Blaxland  
City:: New South Wales 2774  
Country:: Australia

#### CORRESPONDENCE INFORMATION

Correspondence Customer Number:: 05514  
Fax:: (212) 218-2200

#### APPLICATION INFORMATION

Title Line One:: COLOUR CLAMPING  
Total Drawing Sheets:: 30  
Formal Drawings?: Yes  
Application Type:: Utility  
Docket Number:: 169.1867  
Secrecy Order in Parent Appl.?: No

#### REPRESENTATIVE INFORMATION

Representative Customer Number:: 05514

#### PRIOR FOREIGN APPLICATIONS

Foreign Application One:: PQ3775  
Filing Date:: 10-29-1999  
Country:: Australia  
Priority Claimed:: Yes

## COLOUR CLAMPING

### Technical Field of the Invention

The present invention relates to a method and apparatus for resolution conversion of multi-dimensional image data and in particular digital image data. The invention also relates to a computer program product including a computer readable medium having recorded thereon a computer program for resolution conversion of multi-dimensional image data and in particular digital image data.

### Background Art

There are a number of known methods for digital data resolution conversion. For example, transform domain methods such as the fractional Fourier transform (fractional FFT or Chirp-Z transform), the discrete cosine transform (DCT), and discrete wavelet transform (DWT). In addition, there are a number of spatial domain methods such as re-sampling and digital filtering with finite-impulse response (FIR) and infinite-impulse response (IIR) filters and interpolation with continuous, usually cubic, splines. The spline interpolation technique is one interpolation method which can be efficiently implemented using convolution with a continuous kernel. When the continuous kernel produces data that passes through the original data points of an input signal, the kernel is often called an *interpolating* kernel and when the interpolated data produced is not constrained to pass through the original data points it is often called an *approximating* kernel.

Prior art continuous kernels for digital data resolution conversion include the nearest neighbour (NN), linear, quadratic, and cubic kernels. The NN kernel is the simplest method, which interpolates the image with the pixel value that is spatially nearest to the required one. The NN kernel works quite well when the scaling ratio is an integer multiple of the original data as it introduces no new values (ie., no new colours), and preserves sharp edges. However, at other ratios the NN kernel has the disadvantage of shifting edge locations which often produces visible distortions in the output image, especially in images containing text or fine line details. Linear interpolation on the other hand allows for the introduction of new grey levels (or colours) that are effectively used to position edges at sub-pixel locations. However, the Linear interpolation method has the disadvantage of making sharp edges which can appear to be blurred. Quadratic and cubic interpolation provide steeper step responses and therefore less edge blurring, however, the steeper response results in an overshoot on either side of the edge. These overshoots can make the edges in natural images appear sharper, but on text, fine lines, or

other computer generated graphics these overshoots are clearly visible and detract from the perceived image quality and text legibility.

In view of the foregoing discussion, it can be concluded that each kernel has its own strengths and weaknesses, ie., there are certain image areas which are best interpolated using kernels of different shapes. Simply applying a single continuous convolution kernel at every image pixel will not satisfy all of the requirements for a general-purpose resolution conversion application.

The publication United States Patent No. 5,131,057 discloses a kernel with both a steep step response, but no overshoot whereby the parameters of the cubic kernel are adjusted according to image information. US Patent 5,131,057 discloses a two-parameter Catmull-Rom cubic kernel of the form:

$$h(s) = \begin{cases} (2 - \frac{3}{2}b - c)|s|^3 + (-3 + 2b + c)|s|^2 + (1 - \frac{1}{3}b), & |s| \leq 1 \\ (-\frac{1}{6}b - c)|s|^3 + (b + 5c)|s|^2 + (-2b - 8c)|s| + (\frac{4}{3}b + 4c), & 1 < |s| \leq 2 \\ 0, & \text{Otherwise} \end{cases} \quad (1)$$

Choosing parameters  $b$  and  $c$  ( $b = 0$ ,  $c = 0.5$ ), results in an interpolating cubic that agrees with the first three terms of the Taylor series expansion of the original image, and choosing ( $b = 1$ ,  $c = 0$ ) results in the approximating cubic B-spline. In US Patent 5,131,057 the parameter  $b$  is fixed at  $b = 0$ , whilst  $c$  is varied between 0, 0.5, and 1 dependent upon the edge strength measured using a Laplacian of Gaussian (LOG) edge detector. At a sharp edge  $c = 0$  the resulting cubic is:

$$h(s) = \begin{cases} 2|s|^3 - 3|s|^2 + 1, & |s| \leq 1 \\ 0, & \text{Otherwise} . \end{cases} \quad (2)$$

There is however, a problem with using the kernel disclosed in US 5,131,057 to interpolate image data when the re-sampled pixel displacement is not significantly different from the original pixel displacement, say a re-sampling ratio of 10/11 or 11/10. In that instance, pixels at the edges of text and other fine lines take on a grey value rather than the original black or white values resulting in the blurring of sharp edges and a reduction in the observed image quality.

A further problem with conventional continuous convolution kernels is associated with their application to edges at oblique orientations in an image plane. Conventional kernels can be either applied in a separable fashion, ie., first to the rows of the image and then to the columns, or applied in a 2-dimensional form where the kernels are directly convolved with the 2-dimensional image data. However, the kernel orientations in these implementations are limited to either: horizontal, vertical, or symmetrical. Upon encountering an oblique edge, the pixels on either side of an edge are primarily used in the interpolation rather than pixels along the edge resulting in an interpolated edge that no longer appears to be clean and smooth, but appears to be jagged, or blurred, or both. The publication US Patent No. 5,446,804, discloses a method which prevents interpolation across edges by using extrapolated estimates of pixel values for pixels on the other side of the edge in the bilinear interpolation. However, the method disclosed by US Patent No. 5,446,804 requires highly accurate sub-pixel edge estimation at the output resolution and iterative post-processing using a successive approximation procedure, which place high demands on memory and processing resources.

The publication US Patent No. 5,125,042 discloses a method for selecting interpolation kernels based on edge strength, or user input. However, there are some defects that prevent the method disclosed by US Patent No. 5,125,042 from working optimally. Firstly, the use of edge strength alone as the basis for kernel selection does not provide sufficient information for reliable kernel selection (especially at oblique edges). Secondly, Kernel selection based solely upon user input is impractical and does not specify the kernel selection in enough detail. For example, in the sub-image shown in Fig. 7 of U.S. Patent No. 5,125,042, there is not one single kernel that is ideal for the whole sub-image. In general, different kernels are required at a resolution that is impractical to be specified by a user.

Kernel selection needs to be both automatic and reliable so that the appropriate kernel can be used on different edge strengths and edge directions that are typically found in images.

It is an object of the present invention to substantially overcome, or at least ameliorate, one or more disadvantages of existing arrangements.

#### **Disclosure of the Invention**

According to one aspect of the present invention there is provided a method of clamping the output values of filtered image data comprising a mapping of discrete sample values, said method comprising the steps of:

for each discrete sample value of said mapping:

determining a maximum sample value and a minimum sample value of a plurality of input discrete samples values used to calculate said discrete sample value; and

clamping the output value of said discrete sample value to the domain of said plurality of input discrete sample values utilising said maximum sample value and said minimum sample value, wherein said output value is dependent on a plurality of attributes of said plurality of input discrete sample values.

According to another aspect of the present invention there is provided a method of interpolating image data comprising a plurality of discrete sample values, said method comprising the steps of:

accessing at least one portion of said discrete sample values of said image data; calculating kernel values for each discrete sample value of said portion using one of a plurality of kernels;

convolving said kernel values with said portion of discrete sample values to produce an output value; and

clamping said output value to the domain of said portion of discrete sample values, wherein said output value is dependent on a plurality of attributes of said portion of discrete sample values.

According to still another aspect of the present invention there is provided a method an apparatus for clamping the output values of filtered image data comprising a mapping of discrete sample values, said apparatus comprising:

means for determining for each discrete sample value of said mapping, a maximum sample value and a minimum sample value of a plurality of input discrete samples values used to calculate said discrete sample value; and

means for clamping the output value of said discrete sample value to the domain of said plurality of input discrete sample values utilising said maximum sample value and said minimum sample value, wherein said output value is dependent on a plurality of attributes of said plurality of input discrete sample values.

According to still another aspect of the present invention there is provided an apparatus for interpolating image data comprising a plurality of discrete sample values, said apparatus comprising:

access means for accessing at least one portion of said discrete sample values of said image data;

calculation means for calculating kernel values for each discrete sample value of said portion using one of a plurality of kernels;

convolution means for convolving said kernel values with said portion of discrete sample values to produce an output value; and

clamp means for clamping said output value to the domain of said portion of discrete sample values, wherein said output value is dependent on a plurality of attributes of said portion of discrete sample values.

According to still another aspect of the present invention there is provided a computer readable medium, having a program recorded thereon, where the program is configured to make a computer execute a procedure to clamp the output values of filtered image data comprising a mapping of discrete sample values, said program comprising:

code for determining for each discrete sample value of said mapping, a maximum sample value and a minimum sample value of a plurality of input discrete samples values used to calculate said discrete sample value; and

code for clamping the output value of said discrete sample value to the domain of said plurality of input discrete sample values utilising said maximum sample value and said minimum sample value, wherein said output value is dependent on a plurality of attributes of said plurality of input discrete sample values.

According to still another aspect of the present invention there is provided a computer readable medium, having a program recorded thereon, where the program is configured to make a computer execute a procedure to interpolate image data comprising a plurality of discrete sample values, said program comprising:

code for accessing at least one portion of said discrete sample values of said image data;

code for calculating kernel values for each discrete sample value of said portion using one of a plurality of kernels;

code for convolving said kernel values with said portion of discrete sample values to produce an output value; and

code for clamping said output value to the domain of said portion of discrete sample values, wherein said output value is dependent on a plurality of attributes of said portion of discrete sample values.

### **Brief Description of the Drawings**

Embodiments of the present invention will now be described with reference to the drawings, in which:

Fig. 1 is a flowchart showing a method of image interpolation in accordance with the preferred embodiment of the present invention;



Fig. 2 a flowchart showing the pre-processing and initialisation steps in accordance with the method of Fig. 1;

Fig. 3 is a flowchart showing the allocation of edge strengths to an edge map in accordance with the method of Fig. 1;

5 Fig. 4 is a flowchart showing the application of edge direction filters and subsequent edge angle quantisation in accordance with the method of Fig. 1;

Fig. 5 is a flowchart showing the process of edge direction map cleaning in accordance with the method of Fig. 1;

10 Fig. 6 is a flowchart showing the process of determining how the non-directed edges in accordance the method of Fig. 1;

Fig. 7 is a flowchart showing the creation of a LUT index that forms a kernel selection map in accordance with the method of Fig. 1;

Fig. 8 is a flowchart showing a method of interpolation utilising local colour clamping in accordance with a further embodiment of the present invention;

15 Fig. 9 shows a number output images resulting from the methods shown in Figs. 1 to 8;

Fig. 10 is a schematic block diagram of a general purpose computer upon which the preferred embodiment of the present invention can be practiced;

20 Fig. 11 shows a preferred hardware architecture for implementing the method of Fig. 1 as used in DTV;

Fig. 12 shows the preferred hardware architecture for implementing the image interpolation method of Fig. 1;

Fig. 13 shows the preferred hardware architecture for implementing boxcar filtering in accordance the method of Fig. 1;

25 Fig. 14 shows an architecture used for de-interlacing in accordance with the method of Fig. 1;

Fig. 15 shows the preferred hardware architecture for finding the maximum/minimum of a pixel 3 x 3 neighbourhood in accordance with the method of Fig. 1;

30 Fig. 16 shows the preferred hardware architecture for implementing a single stage maximum/minimum finder in accordance with the method of Fig. 1;

Fig. 17 shows the preferred hardware architecture for determining which kind of edge pixel the pixel in the middle of a 3 x 3 neighbourhood is in accordance the method of Fig. 1;

Fig. 18 shows the preferred hardware architecture for implementing the Edge Direction Calculator of Fig. 12;

Fig. 19 shows the preferred hardware architecture for implementing the Morphological Dilator of Fig. 12;

Fig. 20 shows the preferred hardware architecture for implementing the Morphological Eroder of Fig. 12;

Fig. 21 is a flowchart showing the operation of the Kernel Coefficient Calculator of Fig. 12;

Fig. 22 shows the preferred hardware architecture for interpolating with two dimensional kernels in accordance with the method of Fig. 1;

Fig. 23 shows the preferred hardware architecture for interpolating for kernels with  $45^\circ$  and  $135^\circ$  edges in accordance with the method of Fig. 1;

Fig. 24 shows the preferred hardware architecture for output clamping in accordance with the method of Fig. 8;

Fig. 25 shows a graph of the Kernel parameter  $c$  as a function of Edge Strength versus Scaling Factor in accordance with the preferred embodiment of the present invention; and

Fig. 26 shows a graph of the Kernel parameter  $d$  as a function of Edge Strength versus Scaling Factor in accordance with the preferred embodiment of the present invention.

### Detailed Description including Best Mode

Where reference is made in any one or more of the accompanying drawings to steps and/or features, which have the same reference numerals, those steps and/or features have for the purposes of this description the same function(s) or operation(s), unless the contrary intention appears.

When re-sampling a digital image, smooth regions and edge regions need to be re-sampled differently. A long symmetrical kernel, such as the conventional Catmull-Rom cubic with  $c = 0.5$ , is ideal for smooth image regions. A short kernel, such as the Catmull-Rom cubic with  $c = 0$ , is generally good for edge, corner, or highly textured regions. However, in order to reduce the jaggy effect on oblique edges, edge direction also needs to be taken into account in the interpolation process. Edge direction is important so that the interpolation can smooth along an edge, but not across an edge, keeping the edge contour smooth, whilst the edge transition is kept sharp.

The preferred embodiment is a method of image interpolation that produces images with edge contours that have a perceived improvement in smoothness and

sharpness. The preferred method automatically selects the appropriate interpolation kernel for each image region using edge strength, edge direction, and local edge context information. The preferred method of image interpolation is based on the idea of creating a number of binary images that identify the locations of homogeneous image features.

5 The image regions identified in accordance with the preferred embodiment are: smooth areas with no sharp edge features, and edge areas that are either supported by their local neighbourhood to produce approximately linear edges or edges that are not supported by their local neighbourhood and are isolated or complex (corners or intersecting) edges.

The different regions can then be interpolated differently by varying the parameters of a  
10 single convolution kernel, hereinafter called the *universal cubic* convolution kernel.

Fig. 1 is a flowchart showing the method of image interpolation in accordance with the preferred embodiment of the present invention. The overall method will be described first and then each sub-block of Fig. 1 will be described in more detail. The process begins with a number of pre-processing and initialisation steps, ie., steps 101,

15 103, 107 and 109. At step 101, the input image is read together with a requested scaling factor. If the image is being down-sampled the image is initially (box-car) pre-filtered, at step 103, as required, and the requested scaling factor reduced appropriately. At the next

step 105, the (potentially new) scaling factor is used to build a kernel look-up-table (LUT) of the required interpolation coefficients. The process continues at step 107, where a  
20 luminance image is constructed. At the next step 109, the input image can be de-interlaced if it is found that the input image is interlaced. The process continues at step 111, where a non-linear edge detector is applied to the luminance image and an edge map is created. At step 131, smooth areas are detected and directly interpolated using a convolution kernel in accordance with the preferred embodiment. At the next step 113,

25 the edge map is dilated and cleaned using a morphological process. The process continues at step 115, where the directions of the detected edges are determined. At the next step 117, the detected edges are quantised into four directional binary image maps. The process continues at the next step 119, where the edge direction maps are cleaned and directionally eroded in turn. At the next step 121, an additional map of non-directed

edges is created. The process continues at step 123, where the locations of edges, their  
30 directions, and local support are all used to generate a kernel selection map and the kernel selection map is used to vary the parameters of a universal cubic kernel. Thus, an appropriate interpolation can be carried out at each image location. The interpolation is preferably carried out with local limiting, at step 125, so as to ensure that there are no

35 ringing artefacts introduced. The process continues at step 127, where any required

colour space conversion is undertaken. The process concludes at step 129, where an output image of the correct size and format is produced.

When down-sampling image data at a scaling factor less than  $1/4$  pre-filtering in addition to the cubic convolution is applied to the image data. The pre-filtering is necessary since the cubic convolution kernel is of a fixed size, preferably 4 by 4 coefficients, and therefore only filters 4 pixels along both the rows and the columns of an input image when creating an output pixel. The result is that when down-sampling at a scaling factor less than  $1/4$ , some lines or columns in the input image are not used at all to calculate output pixels. Therefore, these lines or columns are effectively *dropped*, as they have no effect on the output. In order to ensure that all input pixels have an effect on the output image, it is necessary to pre-filter and then down-sample the input image. The pre-filtering and down-sampling process ensures that the cubic convolution kernels alone are never used to down-sample a scaling factor of less than  $1/4$ .

Fig. 2 is a flowchart showing the pre-processing and initialisation steps in accordance with the preferred embodiment of the present invention. The process begins with a decision block at step 201 to check the down scaling ratio. When down-sampling at scaling factors less than  $1/4$  the output image is significantly smaller than the input image and so it is not necessary to use a complex pre-filter, at step 203. The preferred method preferably uses a *box-car* (ie., rectangular), pre-filter that simply averages pixels over the size of the rectangular input window. Table 1 illustrates the size of the preferred box-car pre-filter over the range of scaling factors handled by the preferred embodiment. The size along the rows or the columns can preferably be specified separately.

Scaling Factor (SF)	Box-car pre-filter Size	Down-sample rate
$20 > SF \geq 1/4$	None	None
$1/4 > SF \geq 1/8$	2	$1/2$
$1/8 > SF \geq 1/12$	3	$1/3$
$1/12 > SF \geq 1/16$	4	$1/4$
$1/16 > SF \geq 1/20$	5	$1/5$

Table 1

The process continues at step 205, where a check is carried out to determine if the input image is a YUV image. In the case of a YUV image the Y channel is used directly, while for an RGB image the pseudo luminance channel (Y') is created, at step 207, preferably using a rational approximation to the luminance defined in YUV. The process continues at step 209, where a check is carried out to determine whether the input image is interlaced. If the input image is interlaced, each colour channel (including Y') is de-interlaced, at step 211. The kernel selection map created from the luminance image is used to interpolate all three input channels regardless of whether they are in RGB or YUV format. The use of the kernel selection map on the chrominance channels may seem unnecessary, as the human visual system is less sensitive to spatial changes in chrominance than in luminance and therefore the conventional cubic interpolation would seem to be sufficient. However, for implementation simplicity and to guarantee the generation of no false colours particularly on computer graphics and text images, exactly the same interpolation kernels are used on all the input (RGB or YUV) channels.

#### Edge Detection

Fig. 3 is a flowchart showing the allocation of edge strengths to the *EdgeMap*, edge map cleaning and dilation (steps 111 and 113 of Fig. 1), in accordance with the preferred embodiment of the present invention. Edges in the luminance image are detected using a simple pixel difference operator over a 3 by 3 pixel neighbourhood, as shown below:

$$\begin{array}{ccc} Y_6 & Y_7 & Y_8 \\ Y_5 & Y_0 & Y_1 \\ Y_4 & Y_3 & Y_2 \end{array}$$

The process begins at step 302, where, for each centre pixel,  $Y_0$ , the value  $C$  is calculated and compared with a threshold  $T_e$  as follows:

$$C = \max(|Y_0 - Y_i|), \quad i \in 1, \dots, 8 \quad (1)$$

where  $i$  is the index of the eight nearest neighbour pixels of the centre pixel,  $Y_0$ . At the next step 304, a check is carried out to determine if the value of  $C$  is greater than or equal to the threshold,  $T_e$ . If the value of  $C$  is greater than the threshold,  $T_e$ , the pixel  $Y_0$  is labelled as an edge pixel. The process continues at steps 306 to 318, where the allocation

of edge strengths to the *EdgeMap* is carried out. The allocation of edge strengths to the *EdgeMap* is preferably as follows:

$$EdgeMap = \begin{cases} 1 & T_{e1} > C \geq T_e \\ 2 & T_{e2} > C \geq T_{e1} \\ 3 & C \geq T_{e2} \\ 0 & otherwise \end{cases} \quad (2)$$

The process shown in Fig. 3 acts as a non-linear edge detection filter, which reliably detects all of the edges in an image. The filter has been found to be superior to both odd length linear filters, which have a zero response to single pixel wide edges, and even length filters, which effectively measure edge strength mid-way between the original pixels. The preferred threshold values in accordance with the present invention are  $T_e = 64$ ,  $T_{e1} = 128$ , and  $T_{e2} = 192$ . The filter detects pixel differences that are perceived as edges in the image, whilst also detecting more smoothly varying intensity gradients. For the following edge map cleaning process described below any non-zero entry in the *EdgeMap* is effectively treated as an edge during the morphological operations.

#### Edge Detection Cleaning

When interpolating an edge region a short interpolation kernel will often be used (eg., 2 by 2), whilst on a non-edge (smooth) region a longer interpolation kernel will be used (eg., 4 by 4). To ensure that the long kernel used in the smooth regions is not contaminated by edge pixels it is necessary to mark the two pixels adjacent to an edge as edge pixels. The preferred embodiment of the present invention uses a morphological *dilation*, at step 320, with a 3 by 3 structuring element as follows:

$$Se\_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In addition, to remove small isolated edges, that can be due to noise in the image, and to fill in any areas that adjoin two adjacent edge areas, a morphological *opening* (an erosion followed by a dilation) followed by a *closing* (a dilation followed by an opening)

is carried out on the image, at steps 322 and 324, respectively. The opening and closing is preferably carried out with a 4 by 4 structuring element as follows:

$$Se_{-4} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$$

5

### Edge direction detection

The directions associated with the edges detected in the *EdgeMap* are determined at step 115 of Fig. 1. Fig. 4 shows step 115 in more detail with the application of edge direction filters and subsequent edge angle quantisation. The directions are detected using two 1 dimensional filters applied separably to measure edge responses in both the horizontal and vertical directions, at steps 401 and 402, respectively. The filters preferably consist of a low-pass interpolating filter and a high-pass first derivative filter. When the high-pass filter is applied along the rows of an image and the low-pass down the columns, the edge strength in the horizontal direction is calculated and is referred to as  $G_h$ . When the low-pass filter is applied along the rows and the high-pass down the columns, the edge strength in the vertical direction is calculated and is referred to as  $G_v$ . In a further embodiment the filters used to detect the edges are 3-tap optimal edge detectors. In still a further embodiment the edge detection filters are rational approximations to the above optimal filters with only a minimal reduction in performance. The coefficients used in the first embodiment for forming  $G_h$  and  $G_v$  are shown in Table 2. It should be noted that the rational approximation filters are equivalent to the *Sobel* operators commonly used for edge detection.

Optimal Coefficients			
Low-pass (LP)	0.223755	0.552490	0.223755
High-pass (HP)	-0.453014	0	0.453014
Rational Approximation Coefficients			
Low-pass (LP)	1/4	1/2	1/4
High-pass (HP)	-1/2	0	1/2

25

**Table 2**

At the next step 403, the edge direction,  $G_\theta$  is calculated from the edge strengths in the vertical and horizontal directions using:

$$G_\theta = \tan^{-1} \frac{G_v}{G_h} \quad (3)$$

The process continues at steps 405 to 419, where the edge direction,  $G_\theta$  is quantised into one of the four directions: horizontal (0), diagonal ( $\pi/4$ ), vertical ( $\pi/2$ ), and anti-diagonal ( $3\pi/4$ ). The preferred edge angle quantisation in accordance with the preferred embodiment is shown in Table 2. In a further embodiment the inverse tangent and angle quantisation is replaced with a single table look up operation indexed on  $G_h$  and  $G_v$ .

Directional Channel	Edge Angle Quantisation
HorizEdge (0)	$((G_\theta >= -\pi/20) \& (G_\theta <= \pi/20)) \mid ((G_\theta > 19\pi/20) \mid (G_\theta < -19\pi/20))$
DiagEdge ( $\pi/4$ )	$((G_\theta > \pi/20) \& (G_\theta <= 9\pi/20)) \mid ((G_\theta < -11\pi/20) \& (G_\theta >= -19\pi/20))$
VertEdge ( $\pi/2$ )	$((G_\theta > 9\pi/20) \& (G_\theta <= 11\pi/20)) \mid ((G_\theta < -9\pi/20) \& (G_\theta >= -11\pi/20))$
AntidiagEdge ( $3\pi/4$ )	$((G_\theta > 11\pi/20) \& (G_\theta <= 19\pi/20)) \mid ((G_\theta < -\pi/20) \& (G_\theta >= -9\pi/20))$

Table 2

#### Edge Direction Cleaning

Fig. 5 shows the process of edge direction map cleaning in accordance with the preferred embodiment. After the four binary maps corresponding to each of the four edge directions have been created they require cleaning to remove small regions and connect adjoining regions. The cleaning is carried out using binary morphological operations: a morphological *opening* followed by a *closing*, at steps 501 and 503, respectively. In accordance with the preferred embodiment both operations are carried out with a (2 by 2) structuring element as follows:

$$Se\_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}.$$

At the next step 505, a morphological *erosion* operation is carried out. The erosion operation is carried out in order to reduce any edge fading or overshoots and to ensure that directional interpolation is only carried out on edge pixels that are adjacent to other edge pixels of the same direction. The morphological *erosion* operation marks the



centre pixel as non-zero only when all non-zero elements in the structuring element align with non-zero elements in the image. In the case of the horizontal edge map image, the following structuring element is preferably used:

$$Se\_Horizontal = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}.$$

Using the above structuring element, there will be a non-zero output in the horizontal edge map *only* when a horizontal edge pixel has a horizontal edge pixel both to the left and the right of itself. The erosion operation is applied to all the remaining edge direction maps with their appropriate, *directional*, structuring element. For diagonal ( $\pi/4$ ) edges the following structuring element is preferably used:

$$Se\_Diagonal = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix}.$$

For vertical ( $\pi/2$ ) edges the following structuring element is preferably used:

$$Se\_Vertical = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}.$$

Finally, for anti-diagonal ( $3\pi/4$ ) edges the following structuring element is preferably used:

$$Se\_Antidiag = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

### Non-directed Edges

Fig. 6 shows how non-directed edges are determined in accordance with the preferred embodiment. The above erosion operation with each of the directional structuring elements effectively removes pixels from the edge direction maps where edge

pixels are not supported by their neighbours, eg., at a boundary between two edge directions, in corners, or on edges of length less than 3 pixels. The result is pixels that are marked as edges, but are not associated with an edge direction. At step 603, the pixels not associated with an edge direction, determined at step 601, are marked as short edge pixels using the following logical operation:

$$ShortEdge = (EdgeMap \& (\overline{HorizEdge} \vee \overline{DiagEdge} \vee \overline{VertEdge} \vee \overline{AntidiagEdge})).$$

#### Kernel Selection Map Generation

Fig. 7 shows the creation of a LUT index that forms a *kernel selection map*, in accordance with the preferred embodiment of the present invention. There are five binary edge maps (*HorizEdge*, *DiagEdge*, *VertEdge*, *AntidiagEdge*, and *ShortEdge*) that indicate the locations of directed and non-directed (ie., short) edges, respectively, for each pixel in the luminance image. Together with the edge strength information (*EdgeMap*) the five binary edge maps form the kernel selection map, which is used to interpolate the input image. The process for each pixel in the luminance image Y begins at step 701, where an *Offset* is initialised. At the next step 703, an index pointer is initialised to the edge strength information for a pixel in the luminance image. The process continues at steps 705 to 723, where the binary edge maps for the respective pixel are associated with an offset value depending on what value the respective binary edge map is set to. At the next step 725, the kernel selection map value for the respective pixel is calculated according to the following formula:

$$KSMap = (Index \times 4) + Offset. \quad (1)$$

The kernel selection map effectively defines the appropriate set of parameters to apply to the universal cubic kernel so that an appropriately shaped interpolation kernel is used when generating each output pixel. In accordance with the preferred embodiment the kernel selection map contains indexes into a kernel LUT.

#### Mapping the cleaned kernel selection map to the output resolution:

The nearest neighbour (NN) mapping function is preferably used to obtain the kernel selection map at the required output resolution in accordance with the preferred embodiment of the present invention. NN mapping is used to reduce the computation cost

associated with more complex mapping functions, such as a linear mapping function, without significantly reducing the output image quality.

Apply interpolation kernels based on kernel selection map to obtain output image:

5

After the Kernel Selection Map is generated for each output pixel parameter, as shown in Fig. 7, values of one "universal" kernel are read from the Kernel Selection Map and that "universal" kernel is evaluated for the respective output pixel. The *universal cubic interpolation kernel*, in accordance with the preferred embodiment, is defined as follows:

10

$$h(s_x, s_y)_{0 \leq \theta \leq \pi/2} = \frac{1}{\sqrt{2}} \left\{ h((1 - 2\theta/\pi)s_x + (2\theta/\pi)s_y)_{c=0.5} \cdot h(((2\theta/\pi)s_x + (2\theta/\pi - 1)s_y)\omega(\theta))_{c=0} \right\} \quad (1)$$

$$h(s_x, s_y)_{\pi/2 < \theta < \pi} = \frac{1}{\sqrt{2}} \left\{ h((2\theta/\pi - 1)s_x + (2\theta/\pi - 2)s_y)_{c=0.5} \cdot h(((2\theta/\pi - 2)s_x + (1 - 2\theta/\pi)s_y)\omega(\theta))_{c=0} \right\} \quad (2)$$

15

where,  $\omega(\theta)$  known as the across edge weighting factor is a smooth function constrained to pass through 1 when  $\theta = 0, \pi/2$ , and  $\pi$  and through  $3/2$  when  $\theta = \pi/4$  and  $3\pi/4$ . The weighting factor in accordance with a preferred embodiment is defined as follows:

20

$$\omega(\theta) = \frac{5}{4} - \left( \frac{1}{4} \cos(4\theta) \right) \quad (3)$$

In one dimension, (ie., for either row/column or across/along edge kernels), a modified cubic interpolation kernel,  $h(s)$ , is given by:

25

$$h(s) = \begin{cases} 1, & 0 \leq |s| \leq d \\ \left(2 - \frac{3}{2}b - c\right) \left| \frac{s-d}{1-2d} \right|^3 + (-3+2b+c) \left| \frac{s-d}{1-2d} \right|^2 + \left(1 - \frac{1}{3}b\right), & d < |s| \leq 1-d \\ 0, & 1-d < |s| \leq 1+d \\ \left(-\frac{1}{6}b - c\right) \left| \frac{s-3d}{1-2d} \right|^3 + (b+5c) \left| \frac{s-3d}{1-2d} \right|^2 + (-2b-8c) \left| \frac{s-3d}{1-2d} \right| + \left(\frac{4}{3}b + 4c\right), & 1+d < |s| \leq 2-d \\ 0, & \text{Otherwise} \end{cases} \quad (4)$$

where  $c$  and  $b$  are the two parameters of the Catmull-Rom cubic and  $d$  is the parameter controlling the width of the “dead zone” of the cubic interpolation kernel. Figs. 25 and 26 show graphs of the kernel parameters  $c$  and  $d$ , respectively, as a function of Edge Strength versus Scaling Factor, in accordance with the preferred embodiment of the present invention. These graphed values for  $c$  and  $d$  are exemplary and modifications and/or changes can be made thereto without departing from the scope and spirit of the present invention.

In accordance with the preferred embodiment of the present invention, the kernel selection map contains indices to a kernel LUT, which contains interpolation kernels defined by the user. The LUT allows the user to use a different set of kernels for different scaling factors.

In another embodiment of the present invention, the scaling factor is specified in terms of a rational fraction. For example, converting a 640x480 VGA image to a 1024x768 XVGGA image requires a scaling factor of 1.6, which as a fraction is 8/5. The numerator of the re-sampling fraction is called the *up-sample* rate and the denominator the *down-sample* rate. Since the sample distances used to calculate coefficients of the universal cubic kernel, or any other continuous kernel, repeat every  $k$  output pixels, where  $k$  is the down-sample rate, the kernel coefficients required are preferably pre-calculated for a given down-sample rate and preferably stored in a LUT.

After a correct set of 4 by 4 coefficients are evaluated (or read from the kernel LUT,) for a kernel specified by the *kernel selection map* and for horizontal and vertical distances, the co-efficients are convolved with the corresponding set of 4 by 4 input pixels.

#### Down sampling offset calculation

In accordance with a further embodiment of the present invention, optimal horizontal and vertical sampling offsets are calculated to make sure that the output sample points never land on any input sample point when down sampling. The preferred method of the further embodiment adds an offset of 1/2 the sampling distance to the vertical and horizontal sample distances when calculating the kernel coefficients. However, if an output grid point is exactly in the middle of two input grid points, the offset is calculated to be half the distance of the two output grid points. The offset ensures that the interpolated values never pass through the original points when down-sampling, and therefore low-pass filtering is always applied. The preferred method in accordance with the further embodiment significantly improves the performance of cubic interpolation at

rational down-sampling factors such as 1/2, 1/3, 1/4, etc and offers a small improvement in performance at other down-sampling ratios.

5           The optimum offsets are preferably calculated in accordance with the following C code:

```
/* Doubling the upsampling and downsampling ratios so the * overall scaling ratio,
xup/xdn and yup/ydn, remain the * same.
10 */
   ydn*=2;
   yup*=2;
   xdn*=2;
   xup*=2;
15 /* Now calculate the offset*/
   yoff=0;
   if(ydn>yup) { /* if downsampling in y direction
               if (yup%4==0)/* output grid point in the middle of two *input grid points.*/
                   yoff=1;
20       else /* no output grid point in the middle of two input * grid points*/
                   yoff=yup/2;
   }
   xoff=0;
   if(xdn>xup) { /*if downscaling in x direction*/
25       if (xup%4==0)/* output grid point in the middle of two *input grid points.*/
           xoff=1;
       else /* no output grid point in the middle of two input * grid points*/
           xoff=xup/2;
   }
30
```

#### Local Output Clamping

Fig. 8 shows a flowchart showing a method of interpolation utilising local colour clamping in accordance with still a further embodiment of the present invention. In  
35 accordance with the further embodiment, interpolation with the universal cubic kernel

requires a set of (4 by 4) kernel coefficients and the corresponding set of 4 by 4 input pixel values. To reduce the effects of any ringing (ie., over-shoots or under-shoots on sharp edges), that may be produced during the interpolation process of the preferred embodiment, a method of local output clamping can be applied during the interpolation.

5 The process begins at step 801, where the local maximum and minimum value of the input pixels in all of the colour channels, up to  $3 \times 16$  pixels are calculated. The following steps are then carried out for each output pixel  $O_p$  in each colour plane. At the next step 803, values of one "universal" kernel are read from the Kernel Selection Map. The process continues at step 805, where the  $4 \times 4$  co-efficients are read from the kernel LUT and the "universal" kernel is evaluated for the respective output pixel, or read from the kernel LUT. At the next step 807, the corresponding 4 by 4 input pixel values are accessed. The process continues at the next step 809, where the (4 by 4) input pixel values are convolved with the corresponding kernel. At the next step 811, the number of colours in the set of 4 by 4 input pixel colour values are also counted. If the number of  
10 colours in the set is smaller than a certain threshold and if the interpolated output pixel value is larger than the local maximum, then the output value is replaced with the value of the local maximum, at step 813. The preferred threshold is equal to 8, which detects edges adjacent to smooth regions where any ringing artefacts are most visible. Alternatively, the process continues to step 815 and if the number of colours in the set is  
15 smaller than the threshold and if the output value calculated is less than the local minimum, then the output value is replaced with the local minimum, at step 817. Otherwise, the output is clamped between the maximum and minimum pixel values for the whole image. In accordance with the further embodiment, the local maximum and minimum are calculated over all the colour channels resulting in the prevention of  
20 coloured ringing, or colour bleeding, around sharp edges of one (primary) colour.

The method of colour clamping described above can be applied to any filtering process. For example, the process of colour clamping described above can be applied to an image sharpening (ie., enhancement) process.

### 30 Example

In order to illustrate the description of the preferred embodiments discussed above, the sub-images of Fig. 9 show an output resulting from each of the processes. Fig. 9(a) shows an original "Eudora mailbox" image. Fig. 9(b) shows an *EdgeMap* image, as a binary image for  $C > T_e$ , created from the original "Eudora mailbox". The non-linear  
35 filter is shown to reliably detect all salient edges in the image. Fig. 9(c) shows the

*EdgeMap* image after morphological dilation with *Se\_3*. The dilation extends the boundaries of the *EdgeMap* so that pixels less than two pixels away from an actual edge are marked as an edge. Fig. 9(d) shows the *EdgeMap* image after morphological opening and closing with *Se\_4*. The opening and closing operations remove any edge areas less than 4 by 4 in size, and fill non-edge areas less than 4 by 4 in size, respectively. Therefore, small edge areas that could be due to noise and filling or connecting adjacent edge areas are removed. Fig. 9(e) shows the horizontal edges contained in the cleaned *EdgeMap* and the quantisation of the edge angles, calculated by measuring edge strengths in horizontal and vertical directions, as defined in Table 5. Fig. 9(f) shows the diagonal edges contained in the cleaned *EdgeMap*. Fig. 9(g) shows the vertical edges contained in the cleaned *EdgeMap*. Fig. 9(h) shows the anti-diagonal edges contained in the cleaned *EdgeMap*. Fig. 9(i) shows the *HorizEdge* image, ie., areas in the cleaned *EdgeMap* that contain horizontal edges which are supported by their local neighbours. The horizontal edges are shown *after* cleaning with a morphological opening and closing and directional erosion to remove edge pixels not adjacent to edges of the same direction has been carried out. Fig. 9(j) shows the *DiagEdge* image, ie., areas in the cleaned *EdgeMap* that contain diagonal edges which are supported by their local neighbours. Fig. 9(k) shows the *VertEdge* image, ie., areas in the cleaned *EdgeMap* that contain vertical edges which are supported by their local neighbours. Fig. 9(l) shows the *AntidiagEdge* image, ie., areas in the cleaned *EdgeMap* that contain anti-diagonal edges which are supported by their local neighbours. Fig. 9(m) shows the *ShortEdge* image, ie., areas in the cleaned *EdgeMap* that contain short non-supported edges. Short edges are defined to be edges in the *EdgeMap* that are not contained in either the horizontal, diagonal, vertical, or anti-diagonal edge maps. Fig. 9(n) shows the kernel selection map indicating which interpolation kernel is used in which area of the original image. Colour assignments are as follows:

- Black = smooth (*Edgemap* = 0);
- White = non-directed edges (*ShortEdge* = 1);
- Red = horizontal edges (*HorizEdge* = 1);
- Yellow = diagonal edges (*DiagEdge* = 1);
- Blue = vertical edges (*VertEdge* = 1);
- Green = anti-diagonal edges (*AntidiagEdge* = 1);

Note: For simplicity the four different edge strengths are not shown in the final kernel selection map, ie., the final kernel selection map is effectively a kernel selection map with only two edge strengths (*EdgeMap* = [0, 1]).

The aforementioned preferred methods comprise a particular control flow. There are many other variants of the preferred methods which use different control flows without departing from the spirit or scope of the present invention. Furthermore one or more of the steps of the preferred methods can be performed in parallel rather sequentially.

The method of image interpolation is preferably practiced using a conventional general-purpose computer system 1000, such as that shown in Fig. 10 wherein the processes of Figs. 1 to 8 can be implemented as software, such as an application program executing within the computer system 1000. In particular, the steps of the method of image interpolation are effected by instructions in the software that are carried out by the computer. The software can be divided into two separate parts; one part for carrying out the image interpolation methods; and another part to manage the user interface between the latter and the user. The software can be stored in a computer readable medium, including the storage devices described below, for example. The software is loaded into the computer from the computer readable medium, and then executed by the computer. A computer readable medium having such software or computer program recorded on it is a computer program product. The use of the computer program product in the computer preferably effects an advantageous apparatus for image interpolation in accordance with the embodiments of the invention.

The computer system 1000 comprises a computer module 1001, input devices such as a keyboard 1002 and mouse 1003, output devices including a printer 1015 and a display device 1014. A Modulator-Demodulator (Modem) transceiver device 1016 is used by the computer module 1001 for communicating to and from a communications network 1020, for example connectable via a telephone line 1021 or other functional medium. The modem 1016 can be used to obtain access to the Internet, and other network systems, such as a Local Area Network (LAN) or a Wide Area Network (WAN).

The computer module 1001 typically includes at least one processor unit 1005, a memory unit 1006, for example formed from semiconductor random access memory (RAM) and read only memory (ROM), input/output (I/O) interfaces including a video interface 1007, and an I/O interface 1013 for the keyboard 1002 and mouse 1003 and optionally a joystick (not illustrated), and an interface 1008 for the modem 1016. A storage device 1009 is provided and typically includes a hard disk drive 1010 and a floppy disk drive 1011. A magnetic tape drive (not illustrated) may also be used. A CD-ROM drive 1012 is typically provided as a non-volatile source of data. The components 1005 to 1013 of the computer module 1001, typically communicate via an



interconnected bus 1004 and in a manner which results in a conventional mode of operation of the computer system 1000 known to those in the relevant art. Examples of computers on which the embodiments can be practised include IBM-PC's and compatibles, Sun Sparcstations or alike computer systems evolved therefrom.

Typically, the application program of the preferred embodiment is resident on the hard disk drive 1010 and read and controlled in its execution by the processor 1005. Intermediate storage of the program and any data fetched from the network 1020 may be accomplished using the semiconductor memory 1006, possibly in concert with the hard disk drive 1010. In some instances, the application program may be supplied to the user encoded on a CD-ROM or floppy disk and read via the corresponding drive 1012 or 1011, or alternatively may be read by the user from the network 1020 via the modem device 1016. Still further, the software can also be loaded into the computer system 1000 from other computer readable medium including magnetic tape, a ROM or integrated circuit, a magneto-optical disk, a radio or infra-red transmission channel between the computer module 1001 and another device, a computer readable card such as a PCMCIA card, and the Internet and Intranets including email transmissions and information recorded on websites and the like. The foregoing is merely exemplary of relevant computer readable mediums. Other computer readable mediums may be practiced without departing from the scope and spirit of the invention.

The method of image interpolation can alternatively be implemented in dedicated hardware such as one or more integrated circuits performing the functions or sub functions of Figs. 1 to 8. Such dedicated hardware can include graphic processors, digital signal processors, or one or more microprocessors and associated memories. A preferred hardware architecture for carrying out the image interpolation method in accordance with the above-described embodiments will now be described.

Fig. 11 shows the preferred hardware architecture 1103 (hereinafter the 'RC Chip'), as preferably used in a DTV system. An MPEG decoder 1102 accepts compressed video image data from an external source (not illustrated) and decompresses the data. For example, the processor 1005 of the computer system 1000 can supply compressed video image data to MPEG decoder 1102. The image data, in either RGB or YUV format, is then fed into the RC Chip 1103 to perform de-interlacing, frame rate conversion, spatial re-sampling and colour compensation if required. The RC Chip 1103 can store a frame of the input image in embedded DRAM 1101 or some external SDRAM frame buffer 1105. The output image is then fed into a Surface Emission Display (SED) 1107 if the output is in a standard definition (SD), or a high-definition Plasma Display

(PDP) 1109, if the output is in high definition. Both the MPEG decoder 1102 and the RC Chip 1103 are preferably configured by the host CPU 1111.

Fig. 12 shows the hardware architecture of the RC Chip 1103 for carrying out the image interpolation method in accordance with the preferred embodiment. Before normal operation, the RC chip 1103 needs to be initialised by a serial ROM 1201, or by the host CPU 1111 via the a PCI interface 1200. Any suitable microprocessor interface can be used in the RC Chip 1103. The host CPU 1111 writes the kernel coefficients into the Kernel Coefficient Tables 1227, the output width and height and the output time step into output image information, and sets up various registers in all the modules 1203 to 1255 of the RC Chip 1103 via the PCI interface 1200.

When pixel data arrives at the RC chip 1103, a Video Interface Controller (VIC) 1205 accepts the pixel data and writes the data into embedded frame buffers 1203. Alternatively, the frame buffers 1203 could be external. At the same time, pixels are read out of the embedded frame buffers 1205 at the system clock rate. The pixels then go to either interpolation modules (1207, 1227, 1229 and 1231), or edge detection modules (1209 to 1225). An Edge Strength Calculator (ESC) 1211 calculates the edge strength of all input pixels using its 3x3 neighbourhood, while an Edge Direction Calculator (EDC) 1209 determines the direction of edges where they exist. The outputs of both the Edge Strength Calculator 1211 and Edge Direction Calculator 1209 go through some morphological cleaning operations in morphological modules 1211, 1217, 1219, 1221, 1223 and 1225. The results of the edge detection are combined in a Kernel Selector (KS) 1213 and stored in a Kernel Selection Map (KSM) 1215.

After the Kernel Selection Map 1215 is generated, an Interpolation Line Store Manager (ILSM) 1207 begins to read pixels from the frame buffer 1203. A Kernel Coefficients Calculator (KCE) 1229 reads the kernel type from the Kernel Selection Map 1215, and using that information the KCC 1229 it reads kernel coefficients from the Kernel Coefficients Tables 1227. The kernel coefficients are then multiplied with the input image pixel in the Interpolator 1231. The output of the Interpolator 1231 is converted into another colour space in a Colour Space Converter 1233 if necessary. An Output Synchroniser 1235 collects one line of output pixels from the Colour Space Converter (CSC) 1233 and outputs them at the rate of an output pixel clock (not illustrated). An Output Timing Signal Generator 1237 generates an output line sync and page sync signal, as required by the output display.

The following paragraphs describe in detail how the various blocks in Fig. 3 are implemented to perform the functionality described above. The following paragraphs also

include a discussion of some of memory requirements and clock rate required for the preferred hardware architecture.

The Video Interface Controller (VIC) 1205 is responsible for receiving pixel data from an image source (eg. an MPEG decoder). The VIC 1205 writes the pixel data into an embedded (or external) frame buffer 1203 with appropriate addresses. The pixel data is considered valid when both the line sync and page sync signals are asserted at the rising edge of an input pixel clock (not illustrated). The polarities of both line sync and page sync are programmable. The VIC 1205 is synchronous to the input pixel clock. If required, the VIC 1205 can be equipped with Analogue to Digital Converters (ADCs) to convert analog image data into digital form. Filters like a Comb Filter can be added to separate various components from composite signals. When the input image is interlaced, the VIC 1205 stores the image in an interleaved manner. Pixels from the odd field of a frame only go to odd lines of the frame buffer 1203, and even field to even lines.

The VIC 1205 is also responsible for determining the input image width and height. When the VIC 1205 receives the first frame from an image source, the VIC 1205 uses the pixel clock, line sync and page sync signals to count the number of active pixels in a line and the number of active lines in a page. The result of the counts is fed to a Scaling Factor Calculator 1239. During operation the current image width and height are constantly checked against the previous width and height. If a difference is detected, a new scaling factor is calculated to reflect the change. The ability to check the current image width and height and to calculate a new scaling factor automatically optimises the input image to the output display, and allows the user to switch between image sources in a seamless way. In a zoom mode, the VIC 1205 only accepts pixels that are required to produce the output image, which reduces the number of write requests the VIC 1205 generates to a Frame Buffer Controller (FBC) 1241, and the storage required.

The Frame Buffer Controller 1241 is responsible for handling frame buffer 1203 requests from the Video Interface Controller 1205 and a Pixel Extractor 1243. The Frame Buffer Controller 1241 collects read and write requests from both the Video Interface Controller 1205 and a Pixel Extractor 1243, and performs burst read and write operations to the frame buffers 1203. The FBC 1241 contains a DRAM interface (not illustrated) and logic to convert internal bus request signals to DRAM interface signals. The FBC 1241 also has read and write data FIFOs (not illustrated) to collect read and write data. The write circuitry is synchronous to the input pixel clock, and the read circuitry is synchronous to the main system clock (not illustrated).

If the frame buffer 1203 cannot be embedded in the chip, the frame buffer 1203 can be implemented outside the chip and the FBC 1241 can be replaced with an external SDRAM interface (not illustrated). Other functionality of the FBC 1241 remains the same. The FBC 1241 is also responsible for switching frame buffers 1245 and 1247 to implement nearest neighbour frame rate conversion. The FBC 1241 accepts a time step value from a user, which represents the ratio of input frame rate/output frame rate. The FBC 1241 includes a time counter (not illustrated), which is reset to 0 when the RC Chip 1103 is reset and gets incremented by the time step value whenever the RC Chip 1103 outputs a frame. The FBC 1241 determines which input frame in the frame buffer 1203 is to be interpolated next by the content of the time counter.

When the time step value is less than 1 (ie. the input frame rate is slower than the output frame rate), some frames need to be replicated. The next frame buffer to be read is determined by the integer part of the time counter. If the integer part of the time counter (after truncation) is even, the FBC 1241 reads from frame buffer 0 (1245). Otherwise, it reads from frame buffer 1. When the FBC 1241 is reading from one frame buffer, the write circuitry will only write into the other frame buffer.

When the time step value is more than 1 (ie. the input frame rate is faster than the output frame rate), some input frames need to be dropped. When one input frame is written into one of the frame buffers 1245 or 1247, the time counter is incremented with a step value. If the next frame required is not the frame right after the current frame, then the content of that frame will be dropped. The process continues until the required frame arrives. The next frame required is determined by the integer part (after truncation) of the time counter. The FBC 1241 can on average write one pixel and read two pixels in one clock cycle.

The Embedded Frame Buffer 1203 is a 'ping-pong' buffer that stores the current incoming frame in one of its two frame buffers 1245 or 1247, while allowing the rest of the RC chip 1103 to perform resolution conversion on the previous frame in the other frame buffer. The ping-pong buffer switching is described in the following paragraphs. The size of the two frame buffers 1245, 1247 depends on the maximum resolution of the input image. If the maximum resolution of the input image is that of high definition (HD) TV, then the size of the frame buffers is  $2 \times 2048 \times 1080 \times 24$  bits (ie. about 100Mbits) for 3-channel RGB pixels. The size is reduced to 64Mbits if the input is 4:2:0 YUV. Alternatively, the frame buffer can be implemented using external SRAM (eg. PC133 SRAM). Pins need to be allocated for the external SRAM interface.

The access time of the frame buffer 1203 needs to be small enough to support the processing rate of the rest of the logic, which depends on the input pixel rate and the main system clock frequency. If the maximum input rate is from 60Hz interlaced HDTV, then the input pixel clock will be at a rate of about 60MHz. To support an average read of 2  
 5 pixels and A write of 1 pixel per cycle, A memory bandwidth of 180Mpixels/sec is required (ie. 540Mbytes/sec). On a 72 bit-wide memory bus, the above-mentioned memory bandwidth translates to 60 million accesses per second, which can be handled by most known embedded DRAM modules or external PC133 SRAM.

A Scaling Factor Calculator 1239 is responsible for finding the optimum up-  
 10 sampling and down-sampling rates for both horizontal and vertical image resizing. The optimum rates have to satisfy the following conditions, where  $w_{in}$  and  $h_{in}$  are the input width and height, respectively;  $w_{out}$  and  $h_{out}$  are the output width and height;  $up_h$  and  $down_h$  are the optimum horizontal up-sampling and down-sampling rates; and  $up_v$  and  $down_v$  are the optimum vertical up-sampling and down-sampling rates:

- 15 •  $up_h$  is less than or equal to some maximum up-sampling rate;
- $up_h/down_h - w_{out}/w_{in}$  is smallest for all  $up_h$  between 1 and the maximum up-sampling rate,
- $up_v$  is less than or equal to some maximum up-sampling rate; and
- $up_v/down_v - h_{out}/h_{in}$  is smallest for all  $up_v$  between 1 and maximum up-sampling rate.

20 The following pseudo code can be implemented in the Scaling Factor Calculator (SFC) 1239 to find the optimum up-sampling and down-sampling rates in accordance with the preferred embodiment.

```

maxerr = +inf;
up_opt = 0;
25 down_opt = 0;
for (up = 1; up <= maxup; up++) {
    down = up*win/wout;
    down = floor(down);
    err = up*win - down*wout;
30    if (err < maxerr) {
        up_opt = up;
        down_opt = down;
    }
}

```

35

The SFC 1239 is required to deal with the following special cases:

1. When the horizontal or vertical scaling factor is less than  $1/4$ , a boxcar filter section 1249 is used to pre-filter the input image before it is scaled by the Interpolator 1231. As discussed above, Table 1 illustrates the size of a preferred boxcar pre-filter used  
5 over the range of scaling factors handled by the current algorithm. The size along the rows or the columns can be specified separately.

After pre-filtering, the actual horizontal and vertical scaling factors can be modified accordingly. Please note that in accordance with the present invention, when the boxcar filter is used, only cubic interpolation is performed on the pre-filtered image.

10 No edge detection is performed, and no steerable kernels are used. The quality of the image is not decreased by only performing cubic interpolation.

2. When the input image is interlaced and the vertical scaling factor is less than  $1/2$ , de-interlacing is not performed on the image, and the vertical down-sampling is halved instead. Again the quality of the output image is not decreased, and the processing  
15 procedure is greatly simplified. When boxcar filtering is applied on an interlaced image, it should only be applied to pixels in the same field, and no de-interlacing should be needed.

3. When the horizontal-scaling factor is less than 0.75, the horizontal up-sampling rate  $up_h$  and down-sampling rate  $down_h$  are doubled so the actual scaling factor  
20  $up_h/down_h$  remains the same. Doubling the sampling rates allows the offsets to be added to the calculation of the re-sampled output pixel, which prevents any output sample from landing on any original sample. The same procedure is performed when the vertical scaling factor is less than 0.75.

Apart from calculating the optimum horizontal up-sampling and down-sampling  
25 ratios, the SFC 1239 also calculates the optimum horizontal and vertical sampling offsets. These offsets are added to the x and y coordinates of output sample points so that they do not land on any of the input sample points. When the scaling factor (horizontal or vertical) is less than 0.75, the offset is preferably calculated as half the number of up-sample points between two input samples which as discussed above, represents  $1/2$  of the  
30 distance between two input samples. If the offset itself is even, then the offset is further subtracted by 1. The SFC 1239 is implemented to be synchronous to the main system clock.

The Pixel Extractor (PE) 1243 reads pixels out of the frame buffer 1203 row by row, and feeds them to the Edge Detection Line Store Manager(EDLSM) 1209 and the  
35 Interpolation Line Store Manager (ILSM) 1207. The Pixel Extractor 1243 responds to

stall signals from both the line store managers 1209 and 1207 and stops supply of pixels in the clock cycle after the stall signal is asserted. FIFOs (not illustrated) are placed inside the PE 1243 to buffer extra pixels so that THE PE 1243 can perform A burst read from the frame buffer 1203.

When both line store managers 1209 and 1207 are requesting pixels from the PE 1243, they are serviced in a round-robin manner. However, the memory bandwidth from the frame buffer 1203 should be high enough so that the PE 1243 can supply one pixel per clock cycle to both line store managers 1207 and 1209 simultaneously. The maximum pixel rate that the PE 1243 needs to supply to both line store managers 1207 and 1209 is that of the pixel rate of 60Hz interlaced HDTV (ie. 30x2048x1080pixels/sec), about 60Mpixels per second. Therefore, the throughput from the frame buffer 1203 to the PE 1243 should be at least 120Mpixels/sec.

The PE 1243 obtains the input image width and height from the Video Interface Controller 1205 to correctly fetch pixels from the frame buffer 1203. In addition, the PE 1243 keeps track of the locations of the last pixels that are fetched for both line store managers 1207 and 1209. When the input is interlaced, the PE 1243 keeps track of which field it is fetching from, and generates its address to the frame buffer 1203 accordingly.

The PE 1243 is run in synchronous to the main system clock. When the input image is interlaced and the vertical scaling ratio is less than 0.5, the PE 1243 fetches pixels from one field only. So interpolation is performed on pixels in the same field. If the vertical scaling ratio is greater than 0.5, the odd and even fields are preferably interleaved together, ie. one row from the odd field is followed by one row from the even field, and vice versa. The PE 1243 is also responsible for finding the luminance of the input pixels and feeding the luminance to the EDLSM 1209. If the input image is in YUV format, then the PE 1243 only needs to feed the Y channel to the EDLSM 1209. If the input image is in RGB format, then THE PE 1243 creates a luminance channel (Y') using a rational approximation to the luminance defined in YUV, as defined by the following equation:

$$Y' = 1/2G + 1/4R + 1/4B.$$

The Boxcar Filter (BF) section 1249 is responsible for pre-filtering the input image before the input image is further down-sampled by the Interpolator 1231. A boxcar filter of width  $k$  and height  $l$  can be represented by a matrix with  $k$  columns and  $l$  rows, and all coefficients equal to  $1/(k*l)$ . For example, a boxcar filter of dimension 3x3 can be represented by the matrix below.

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

In accordance with the preferred embodiment, a boxcar filter is applied to a block of  $k \times l$  pixels to reduce them to only 1 pixel, as discussed above. The preferred embodiment is different from the conventional methods of applying the filter by sliding the filter across an image. If a boxcar filter of dimension  $k \times l$  is applied to an image of dimension  $m \times n$ , then the output image will be shrunk to a dimension of  $\lceil m/k \rceil \times \lceil n/l \rceil$ . When  $m$  is not a multiple of  $k$ , the image is reflected on the right edge. When  $n$  is not a multiple of  $l$ , the image is reflected on the bottom edge. In accordance with the preferred embodiment, the width and height of the boxcar filter 1249 are determined by the horizontal and vertical scaling factors ( $s$  and  $t$  respectively), as discussed above.

Fig. 13 shows the preferred method of implementing boxcar filtering. The BF 1249 is synchronous to the main system clock. To implement a  $k \times l$  boxcar filter, pixels from the input image are stored in a FIFO 1301. The pixels are read from the other end and put into a shift register 1303. After  $k$  pixels have been read from the FIFO 1301, their sum is calculated by a summing unit 1307, and then stored in the accumulator line store 1305. After the first line of pixels is processed, the second line of pixels are then processed in a similar way. The only difference, however, is that when the  $n^{\text{th}}$  sum of  $k$  pixels is multiplied with the horizontal weight, the result is added with the  $n^{\text{th}}$  value in the accumulator line store 1305. The sum is then written back to the  $n^{\text{th}}$  place in the accumulator line store 1305 as the new accumulator value. Therefore, all the partial sums are accumulated. At the  $l^{\text{th}}$  line of pixels, the accumulator value is multiplied with the weight using a multiplying unit 1309, which should be  $1/k \times l$ , and 0 should be written into the accumulator line store 1305. The product is the final output of the boxcar filter 1249.

In accordance with the preferred boxcar filter, the maximum length of the line store is the maximum image width. Also, the preferred boxcar filter can accept input pixels at every clock cycle. To minimise the logic used, a slow adder 1307 and multiplier 1309 are preferably used. Every addition and multiplication has a minimum of two clock cycles to complete.

The Interpolation Line Store Manager (ILSM) 1207 accepts pixels from the Pixel Extractor 1243 and stores the pixels in the local line stores 1251-1255. The ILSM 1207 is also responsible for sending pixels to the Interpolator 1231. If the input image is interlaced, the ILSM 1207 also de-interlaces the image. There are five line stores 1251-1255 in the ILSM 1207. In the duration of processing one line the ILSM 1207 writes the



pixel from the PE 1243 to one of the line stores 1251. At the same time, the ILSM 1207 reads pixels at the same location as the written pixel from the other four line stores 1252-1255. When the input line store 1251 is full, the line store containing the least recent line is selected to accept the next incoming line. Therefore, the line stores, 1251-1255, contain the most recent lines from PE 1243 to the ILSM 1207.

The ILSM 1207 operates strictly in a one-in-one-out fashion. The ILSM 1207 accepts one pixel per cycle from the PE 1243, and sends four pixels to the Interpolator 1231, unless the stall signal from the Interpolator 1231 is asserted. When the stall signal from the Interpolator 1231 is asserted, the ILSM 1207 stalls sending pixels in the next cycle, and transmits a stall signal back to the PE 1243.

When the input image is interlaced, the ILSM 1207 de-interlaces the image. The preferred embodiment utilises a median filter method for de-interlacing the image. There are other methods of de-interlacing. However, the median filter is the most efficient and easiest to implement. Intra-frame de-interlacing methods can also be implemented using the spatial methods discussed above. In the median filtering method, a missing pixel is reconstructed by finding the median of the pixels above and below the missing pixel, as well as the pixel in the same location in previous frame. Fig. 14 shows an architecture used for de-interlacing in accordance with the preferred embodiment. When a pixel at coordinates  $(x,y)$  is read from the PE 1243, the pixel is stored in one of the line stores 1251-1255. At the same time, the ILSM 1207 reads the pixel just above the incoming pixel in the current field at coordinates  $(x,y-1)$ , and also reads the pixel at the missing location in the last field, from two of the other four line stores 1252-1255. The luminance of these three pixels are fed into 3 comparators 1401, 1403 and 1405, and the outputs of the comparators 1401-1405 control the operation of a 3-to-1 multiplexer 1407, as described in Table 3 below.

A>B	B>C	C>A	Output
0	0	1	B
1	0	0	C
1	0	1	A
0	1	0	A
0	1	1	C
1	1	0	B

Table 3

The output of the de-interlacing architecture, shown in Fig. 14, in accordance with the preferred embodiment is the median of the three pixels. The line stores 1251-1255 are preferably long enough to store the widest image coming into the RC chip 1103. If the widest image comes from HDTV, then each line store 1251-1255 needs to be able to store 2048 3-channel pixels, ie. 3x2048bytes. Therefore, 5 line stores would take 30kbytes. For 4:2:0 YUV input pixels, the size of the line stores is reduced to 20kbytes. Alternatively, when the width of the input image is smaller than the maximum width, the ILSM 1207 can be implemented so that it can store more than 1 line in one line store. Storing more than 1 line in one line store means that the ILSM 1207 can store more than 5 lines with the available memory. The ILSM 1207 is preferably implemented so as to be synchronous to the main system clock.

The Edge Detection Line Store Manager (EDLSM) 1209 accepts luminance pixels from the PE 1243 and stores the pixels in the local line stores 1257-1260. The EDLSM 1209 is also responsible for sending pixels to the Edge Strength Calculator 1211 and Edge Direction Calculator 1225, where edges are detected and their direction calculated. If the input image is interlaced, the EDLSM 1209 also de-interlaces the image if required. There are 4 line stores in the EDLSM 1209. In the duration of processing one line, the EDLSM 1209 writes the pixel from the PE 1243 to one of the line stores 1257. At the same time, the EDLSM 1209 reads pixels at the same location from the other 3 line stores 1258-1260, and sends the pixels to the edge detection modules 1211 and 1225. When the receiving line store is full, the line store containing the least recent line is selected to accept the next incoming line. In accordance with the preferred embodiment, the line stores always contains the most recent lines from the PE 1243 to the EDLSM 1257.

The EDLSM 1257 operates strictly in a one-in-one-out fashion. Generally, the EDLSM 1257 accepts one pixel per cycle from the PE 1243, and sends 3 pixels to the edge detection modules 1211 and 1225, unless the stall signal from the one of these modules 1211 and 1225 is asserted. When that happens, the EDLSM 1257 stalls sending pixels in the next cycle and transmits that stall signal back to the PE 1243. The Edge Strength Calculator 1211 and the Edge Direction Calculator 1225 preferably operate in synchronous with one another, such that the respective stall signals should be the same.

If the input is interlaced, the EDLSM 1257 de-interlaces the image. However, no RGB-to-Y conversion is required because the pixel contains luminance already. The line stores 1257-1260 are preferably long enough to store the widest image coming into the RC chip 1103. If the widest image comes from HDTV, then each line store 1257-1260

needs to be able to store 2048 pixels, ie. 2048bytes. Therefore, 4 line stores would take 8kbytes. Optionally, when the width of the input image is smaller than the maximum width, the EDLSM 1209 can be implemented so that it can store more than 1 line in one line store. Storing more than 1 line in one line store means that the EDLSM 1209 can store more than 4 lines enabling edge detection and angle calculation with larger kernels to be performed. The EDLSM 1209 operates synchronously with the main system clock.

The Edge Strength Calculator (ESC) 1211 is responsible for calculating the edge strength of each input pixel using a non-linear filter on the 3x3 neighbourhood around the pixel. The maximum/minimum of a pixel 3 x 3 neighbourhood can be found by the circuit in Fig. 15. The three columns of pixels are each fed into a maximum/minimum finder 1501 and their respective minimum/maximum are found. The minima of these finders are fed a 3-stage shift register 1507, and the maximum into another shift register 1505. The 3 outputs of the minimum shift register 1507 are fed into another maximum/minimum finder 1503, and the minimum output of that is the minimum of the region. Similarly, the 3 outputs of the maximum shift register 1505 are fed into another maximum/minimum finder 1509, and the minimum output of that is the minimum of the region. The single stage maximum/minimum finders can be implemented with a circuit shown in Fig. 16. The lookup table operation is described in Table 4.

A>B	B>C	C>A	Minimum	Maximum
1	1	0	C	A
0	1	1	A	B
1	0	1	B	C
0	0	1	A	C
1	0	0	B	A
0	1	0	C	B

Table 4

After finding the maximum and minimum of the neighbourhood, the circuit shown in Fig. 17 determines which kind of edge pixel the pixel in the middle is in accordance with the preferred embodiment of the present invention. Part A of the circuit includes an adder 1701, a 'greater than one' section 1703 and a comparator 1705. Part A of the circuit determines whether the centre pixel is closer to the maximum. If the centre pixel is closer to the maximum, then the difference between the centre pixel and the

minimum is used to compare against the thresholds using three comparators 1707, 1709, 1711 and an encoding block 1713. If the centre pixel is closer to the minimum, then the difference between the maximum and the centre pixel is used instead. The output of the comparators 1707, 1709 and 1711 are encoded at the end to give the edge type. The thresholds are preferably 64, 128 and 192. The ESC 1211 is synchronous to the main system clock and is preferably able to determine the edge type of one input pixel per clock cycle.

Fig. 18 shows the architecture of the Edge Direction Calculator (EDC) 1225 in accordance with the preferred embodiment. The EDC 1225 is responsible for finding the directions of edges in the input image. The EDC 1225 applies two 3x3 separable kernels to a column of 3 input pixels and calculates the quotients between the results from the above operations. The quotient is then compared with a set of thresholds to determine the direction. A high pass filter 1803 and a low pass filter 1801 are applied to the pixels. The outputs of the filters 1801 and 1803 are fed into two 3-stage shift registers 1805 and 1806, respectively. The 3 entries of the low pass filter result register 1805 are fed into a high pass filter 1807 to calculate the horizontal edge strength ( $G_h$ ). The 3 entries in the high pass filter result register 1806 are fed into a low pass filter 1809 to calculate the vertical edge strength ( $G_v$ ). The absolute value of the horizontal edge strength is fed into a reciprocal ROM 1811 and the reciprocal of  $G_h$  is multiplied with  $G_v$  to find the quotient. The quotient  $G_v/G_h$  is fed into the angle quantisation logic 1813 to find out which of the 4 directions the particular edge is in. The angle quantisation logic 1813 is made up of a number of comparators (not illustrated) and an encoder (not illustrate) that encodes the results of the comparators into edge type.

The angle quantisation logic can be described by the following C code:

```
G=arctan(Gv/Gh);
if (((G>=-pi/20) && (G<=pi/20)) || ((G>19*pi/20) && (G<-19*pi/20)))
    angle = 0; // horizontal edge
else if (((G>pi/20) && (G<=9*pi/20)) || ((G<-11*pi/20) && (G>=-19*pi/20)))
    angle = 1; // diagonal edge
else if (((G>9*pi/20) && (G<=11*pi/20)) || ((G<-9*pi/20) && (G>=-11*pi/20)))
    angle = 2; // vertical edge
else // if (((G>pi/20) && (G<=9*pi/20)) || ((G<-11*pi/20) && (G>=-19*pi/20)))
    angle = 3; // anti-diagonal edge
```

The EDC 1225 generates one angle type per clock cycle and operates synchronously with the main system clock.

After the edge strength is found by the Edge Strength Calculator 1217, the Morphological Dilator (MD) 1217 dilates the edge map with a 3x3 structuring element before the edge map is morphologically opened and closed. After the edge direction maps are found they are morphologically opened and closed. The edge maps are eroded with their respective 3x3 structuring element by a Morphological Eroder (ME) 1221. Fig. 19 shows the preferred implementation of the MD 1217 with two 1-bit line stores 1901 and 1902. Fig. 20 shows the implementation of the ME 1221 with 2 1-bit line stores 2001 and 2002.

In accordance with the preferred embodiment the edge strength map and the edge direction maps are cleaned by morphological open and close operations. The morphological opening operations fit the structuring element on the image. If all pixels under the 1's in the structuring element are 1 then the 1's can stay. If one of them is 0 then all the other 1's are changed to 0. Conventional methods of fitting the structuring element on the image use morphological dilation and erosion, but this is costly in terms of the number of line stores required. For rectangular structuring elements with all 1's the method can be simplified to a template fitting method that uses a fewer number of line stores. The following Matlab code describes the template fitting algorithm in accordance with the preferred embodiment.

% in opening, if the structuring element fits, then the marks stay

```
for i=1:size(ipad,1)
    pixel_cnt=0;
    for j=1:size(ipad,2)
        if ipad(i,j)==match
            pixel_cnt=pixel_cnt+1;
            if pixel_cnt>se
                pixel_cnt=se;
            end
        else
            pixel_cnt=0;
        end
        if j>=se
            if pixel_cnt==se
                good=1;
```

```

else
    good=0;
end
if i>=se
5      if (good==1) & (ls_in(j-se+1)==se-1)
        ocode(i-se+1,j-se+1)=1;
        elseif (i>se) & (((good==0) & (ls_in(j-se+1)==se-1) & (ls_out(j-
se+1)==se-1)) | ls_out(j-se+1)>0)
        ocode(i-se+1,j-se+1)=1;
10      else
        ocode(i-se+1,j-se+1)=0;
        end
    end
    if good==1
15      if ls_in(j-se+1)==se-1
        ls_out(j-se+1)=se;
        end
        ls_in(j-se+1)=ls_in(j-se+1)+1;
        if ls_in(j-se+1)>se-1
20      ls_in(j-se+1)=se-1;
        end
    else
        ls_in(j-se+1)=0;
    end
25    ls_out(j-se+1)=ls_out(j-se+1)-1;
    if ls_out(j-se+1)<0
        ls_out(j-se+1)=0;
    end
end
30 end
end
end

```

In the above code, 'se' represents the width of the structuring element, which is also the height of the structuring element in this case. In accordance with the preferred embodiment of the present invention se is equal to 4. If the pixel is 1, then the pixel

counter `pixel_cnt` is incremented. If the pixel is 0, the pixel counter is reset. If the pixel counter equals `se`, the sequence of 1's is considered to be 'good'. There are two line stores, `ls_in` and `ls_out`, which stores temporary results. The line store `ls_in` keeps track of where the 'good' sequences start, and the number of lines that have such 'good' sequences of 1's at that position. If the number of lines having 'good' sequences at the same position equals `se`, then the structuring element template fits, and so the output can start to generate 1's in those positions. The line store `ls_out` keeps track of where the logic needs to generate 1's, and the number of lines that need to be generated. At the beginning of the process there are `se` lines, but as a line is generated the number is decremented. When `ls_out` is equal to 0 then 0's are generated in those positions.

In the morphological closing operation in accordance with the preferred embodiment, if the pixels under the 1's in the structuring element are 0, then all 0's can stay. Otherwise, they are all changed to 1. Therefore, similar hardware to that described above for the morphological opening operation can be used for the morphological closing operation, except that the pixels need to compare with 0 instead of 1. After the closing operation there is a decoding stage that transforms the output of the morphological closing operation to a normal pixel sequence. The Matlab code for the decoding stage is as follows:

```
for i=1:(size(ipad,1)-padding)
    for j=1:(size(ipad,2)-padding)
        if (j-se<0)
            j_low=1;
        else
            j_low=j-se+1;
        end
        if (sum(ocode(i,j_low:j))>0)
            ipad(i,j)=match;
        else
            ipad(i,j)=1-match;
        end
    end
end
```

The decoding stage looks at the last 'se' entries in the compressed code. If any of the last 'se' are 1, then the decoding stage generates 1. Otherwise a 0 is generated.

Both the morphological opening and closing operations take 1 clock cycle to generate 1 pixel. However, both operations have pipeline delays of 3 lines for a 4x4 structuring element. Therefore, the whole morphological cleaning operation generates 1 pixel per cycle, with 6 lines of pipeline delay.

5       The Kernel Selector (KS) 1213 combines the edge strength map and the edge direction maps to form the Kernel Selection Map 1215. Firstly, the KS 1213 finds out where short kernels need to be used in the image by the following boolean equation:

$$ShortEdge = (EdgeMap \& (\overline{HorizEdge} \mid \overline{DiagEdge} \mid \overline{VertEdge} \mid \overline{AntidiagEdge})).$$

10       Secondly, the KS 1213 encodes the edge strength and the edge types (smooth, short, horizontal, etc) together to form an index to the Kernel Coefficient Tables 1227. As discussed above, Fig. 7 shows the method for encoding the edge strength and the edge types. The KS 1213 preferably makes a kernel selection per clock cycle and is preferably synchronous to the main system clock.

15       The Kernel Selection Map (KSM) 1215 contains the type of kernel that is preferably used to interpolate for each input pixel. As discussed above, the KSM 1215 is interpolated to output resolution using nearest-neighbour interpolation. Since each kernel type is encoded with 5 bits, the KSM 1215 has to be 2048x5bits (ie. 10kbits). Only the input line that is currently interpolating is needed.

20       As the Kernel Selector 1213 generates edge types for input pixels, the edge types are fed straight to the Kernel Coefficient Calculator 1229 for kernel coefficient fetching, and the Interpolator 1231 for interpolation. The edge types are also stored in the Kernel Selection Map 1215 so that if the next output line uses the same input line, the edge types need not be recalculated. The Kernel Selection Map 1215 supports one read request or one write request in one clock cycle and is synchronous to the main system clock.

25       The Kernel Coefficient Tables (KCT) 1227 contain all the coefficients required for interpolation. There are 4 sections to the KCT 1227: smooth kernels, vertical/horizontal across-the-edge kernels (including short kernels), diagonal/anti-diagonal along-the-edge kernels, and diagonal/anti-diagonal across-the-edge kernels. The actual coefficients are the products of two entries from the KCT 1227. These kernel tables can preferably support 256 up-sample points between 2 input pixels, and each  
30       coefficient is preferably a 16-bit signed 8.8 fixed point number. The KCT 1227 are described in more detail in the following paragraphs.

Each up-sample point 'x' preferably has 4 entries in the smooth kernel table 1227a, representing the kernel coefficients at x-1, x, x+1 and x+2. The total size of the



smooth kernel table 1227a is preferably 4x256x16 bits (ie. 2kbytes) and preferably provides 4 entries in one clock cycle.

There is one group of short kernel tables 1227b for each edge strength. Hence there are 3 groups of short kernel tables. Each group of short kernel tables 1227b has 4 entries for each up-sample point  $x$ , representing the kernel coefficients at  $x-1$ ,  $x$ ,  $x+1$ ,  $x+2$ . The total size of the short kernel tables 1227b is preferably 3x4x256x16 bits (ie. 6kbytes) and preferably provides 4 entries in one clock cycle.

For the diagonal along the edge kernel tables 1227c the number of up-sample points is doubled, since the coefficients are halved. Each group of diagonal along the edge kernel tables 1227c has 8 entries for each up-sample point  $x$ , representing the kernel coefficients at  $x-1.5$ ,  $x-1$ ,  $x-0.5$ ,  $x$ ,  $x+0.5$ ,  $x+1$ ,  $x+1.5$ ,  $x+2$ . The total size of the tables is preferably 8x512x16 bits (ie. 8kbytes) and preferably provide 8 entries in one clock cycle.

There is 1 group of diagonal across -the-edge kernel tables 1227d for each edge strength including 'no edge strength'. The number of up-sample points is doubled, since the coefficients are halved. Each group of diagonal across-the-edge kernel tables 1227d preferably has 8 entries for each up-sample point  $x$ , representing the kernel coefficients at  $k(x-1.5)$ ,  $k(x-1)$ ,  $k(x-0.5)$ ,  $kx$ ,  $k(x+0.5)$ ,  $k(x+1)$ ,  $k(x+1.5)$ ,  $k(x+2)$ , where  $k$  is preferably  $3/2$  in accordance with the preferred embodiment. The total size of the tables is preferably 4x8x512x16 bits (ie. 32kbytes) and preferably provides 8 entries in one clock cycle.

The Kernel Coefficient Calculator (KCC) 1229 is responsible for fetching the appropriate kernel coefficients from KCTs 1227. The KCC 1229 accepts the horizontal and vertical up-sampling rate and down-sampling rate from the Scaling Factor Calculator 1239, as well as the optimum horizontal and vertical offsets in cases of down-sampling. There are two sets of counters (not illustrated) in the KCC 1229 keeping track of the location of the output sample point, one for horizontal coordinate and one for vertical. In each set of counters there is a coordinate counter (not illustrated) counting the  $x$ -coordinate of the input sample point which is to the left and above the output sample point, and there is an offset counter (not illustrated) storing the offset from that input sample to the output sample.

Fig. 21 is a flowchart showing the operation of the KCC 1229 in accordance with the preferred embodiment of the present invention. The process begins at step 2101, where the coordinate counters for the  $x$  and  $y$  coordinates are set to 0, while the offset counters are set to the optimum offset. The following steps then occur each time an output pixel is generated. At the next step 2103, at the beginning of a line, the KCC 1229

uses the content in the y offset counter to fetch the kernel coefficients of the vertical kernel from smooth kernel tables 1227(a) and short kernel tables 1227(b). At the next step 2105, the kernel coefficients are stored in registers (not illustrated) and are used throughout the line. At the next step 2107, if the kernel selected is smooth, the process

5 continues to step 2113, where the KCC 1229 uses the content in the x offset counter to fetch kernel coefficients in the horizontal kernel from the smooth kernel table 1227(a). The type of kernel table that the KCC 1229 fetches from depends on the kernel used to interpolate the output pixel. At the next step 2109, if the kernel selected is horizontal, the process continues to step 2115, where the KCC 1229 uses the content in the x offset

10 counter and the edge strength to fetch kernel coefficients in the horizontal kernel from the smooth kernel table 1227(a). At the next step 2111, if the kernel selected is vertical, the process continues to step 2117, where the KCC 1229 uses the content in the x offset counter to fetch kernel coefficients in the vertical kernel from the short kernel table 1227(b). At the next step 2119, if the kernel selected is diagonal, the process continues to

15 step 2121, where the KCC 1229 calculates the absolute values of the difference of the x and y offsets, and uses the absolute value to fetch kernel coefficients from the diagonal along-the-edge tables. At the next step 2123, the KCC 1229 calculates the absolute value of the sum of the x and y offsets, and uses the absolute value to fetch kernel coefficients from the diagonal across-the-edge tables, along-the-edge tables. If the kernel selected is

20 anti-diagonal at step 2125, the process continues to step 2127, where the KCC 1229 uses the sum of x and y offsets to fetch coefficient values from the along-the-edge tables. At the next step 2129, the KCC 1229 uses the difference of the x and y offsets to fetch coefficient values from the across-the-edge tables. When fetching is finished, the horizontal down-sampling rate is added to the x offset counter, at step 2131. If the x-

25 offset counter is more than the up-sampling rate, at step 2133, then the process continues to step 2135, where the up-sampling rate is subtracted from it, and the x-coordinate counter is incremented. The process is iterative until a whole line of output pixels is produced. At the next step 2137, when a whole line of output pixels is produced, the process continues to step 2139, where the vertical down-sampling rate is added to the y

30 offset counter. At the next step 2141, if the y offset counter is more than the up-sampling rate, then the process continues to step 2143, where the up-sampling rate is subtracted from it, and the y coordinate counter is incremented. The process concludes at step 2145, when all of the output pixels have been produced.

The KCC 1229 is preferably capable of fetching enough coefficients for

35 interpolation of 1 output pixel in one clock cycle meaning that the KCC 1229 should be

able to fetch 4 kernel coefficients for smooth, horizontal, vertical and short kernels, and 14 coefficients for diagonal and anti-diagonal kernels. The KCC 1229 is synchronous to the main clock.

The Interpolator 1231 is responsible for performing convolution on 4x4 input pixels with 4x4 interpolation kernel coefficients to find the output pixels. The Interpolator 1231 accepts the pixels from the ILSM 1207 and the kernel coefficients from the KCC 1229, and produces resultant output pixels to the Colour Space Converter 1233. The Interpolator 1231 includes separate hardware (shown in Fig. 22) for interpolating with both 2D-separable kernels (ie., smooth, short, vertical and horizontal kernels) and 45° and 135° edge kernels (shown in Fig. 23 below). The Interpolator 1231 also includes hardware (shown in Fig. 24 below) for Output Clamping.

Fig. 22 shows the Interpolation hardware for 2D-separable kernels in accordance with the preferred embodiment of the present invention. To generate the  $n^{\text{th}}$  output pixel in a line, the value of  $p$  is determined. The value  $p$  represents the smallest integer greater than the  $n \times \text{horizontal\_step}$ . The interpolator hardware of Fig. 22 fetches the  $(p-2)^{\text{th}}$ ,  $(p-1)^{\text{th}}$ ,  $p^{\text{th}}$  and  $(p+1)^{\text{th}}$  pixels from 4 line stores 1251 to 1255 in the ILSM 1207. The incoming pixels are preferably stored in line stores 1251, 1252, 1253 and 1254 in that order, so that the last 4 most recent lines are in these line stores. The incoming stream of pixels will be stalled if generation of the next line of pixels does not need an extra line of pixels from the input. Two of the line stores can be shared with the line stores used in edge detection. The  $(p-2)^{\text{th}}$ ,  $(p-1)^{\text{th}}$ ,  $p^{\text{th}}$  and  $(p+1)^{\text{th}}$  pixels are multiplied with vertical smooth kernel coefficients  $Y^0_{\text{smooth}}$ ,  $Y^1_{\text{smooth}}$ ,  $Y^2_{\text{smooth}}$ , and  $Y^3_{\text{smooth}}$ , and vertical short kernel coefficients  $Y^0_{\text{short}}$ ,  $Y^1_{\text{short}}$ ,  $Y^2_{\text{short}}$ , and  $Y^3_{\text{short}}$ , using the multipliers 2201-2208.  $Y^0_{\text{smooth}}$ ,  $Y^1_{\text{smooth}}$ ,  $Y^2_{\text{smooth}}$ ,  $Y^3_{\text{smooth}}$ ,  $Y^0_{\text{short}}$ ,  $Y^1_{\text{short}}$ ,  $Y^2_{\text{short}}$ , and  $Y^3_{\text{short}}$  can be fetched at line sync time, so they only need a register. For each kernel, the sum of the 4 products is found using the adders 2209 to 2214 and stored in one of two shift registers 2215 and 2217. The outputs of the shift registers 2215 and 2217 are fed to a set of multiplexers 2219-2222 which are controlled by a signal 'smooth'. The signal 'smooth' is 1 when the current interpolation kernel is a smooth kernel or a vertical kernel. The outputs of the multiplexers 2219-2222 are multiplied with  $X^0$ ,  $X^1$ ,  $X^2$ , and  $X^3$  respectively, using multipliers 2223 to 2229. The products are then added together, using adders 2231 to 2235, to find the interpolated pixel.  $X^0$ ,  $X^1$ ,  $X^2$  and  $X^3$  vary from one cycle to another, therefore they need to be calculated at every cycle. However, if the horizontal step value can be expressed in terms of  $p/q$ , where  $q$  is reasonably small, then in total only  $q$  sets of  $\{X^0, X^1, X^2, X^3\}$  are needed. These sets correspond to  $s$  equal to 0,  $1/q$ ,  $2/q$ , .....,  $(q-1)/q$ .

The sets of  $\{X^0, X^1, X^2, X^3\}$  are values which are fixed for the whole image, so they can be calculated at the start-up time and stored in a LUT.

Depending on the horizontal step size, some or all of the results from vertical interpolation used to generate the next output pixel can be reused, as the horizontal step size is always less than 4. As a result, the Interpolator 1231 shown in Fig. 22 preferably accepts one input pixel per cycle when scaling down an image and the Interpolator preferably outputs one pixel per cycle when scaling up an image. The Interpolator 1231 of Fig. 22 does not work for kernels for  $45^\circ$  and  $135^\circ$  edges, since in these kernels both multipliers have x and y coordinates in them.

Fig. 23 shows the interpolation hardware to interpolate for kernels with  $45^\circ$  and  $135^\circ$  edges, in accordance with the preferred embodiment. To generate the  $n^{\text{th}}$  output pixel in a line, the value of  $p$  is determined. The value  $p$  represents the smallest integer greater than  $n * \text{horizontal\_step}$ . The interpolator hardware of Fig. 23 fetches the  $(p-2)^{\text{th}}$ ,  $(p-1)^{\text{th}}$ ,  $p^{\text{th}}$  and  $(p+1)^{\text{th}}$  pixels from 4 line stores 1251 to 1255 in the ILSM 1207 and multiplies them, using multipliers 2301 to 2316, with the kernel coefficient values generated from the difference of the offsets in the x and y directions. The sum of the products in all of the columns are found using the adders 2321. All of the partial sums are then multiplied, using multipliers 2322 to 2328, with the kernel coefficients generated from the sum of the offsets in the x and y directions. The adders 2329 to 2331 are then used to add the products together to find the interpolated pixel. Extra pipeline registers (not illustrated) may have to be added to increase the frequency at which the interpolator hardware of Fig. 23 can operate. The hardware of Fig. 23 preferably takes 1 cycle to produce an output pixel, as long as one set of kernel coefficients and pixels are supplied in every clock cycle. The hardware of Fig. 23 preferably uses 4 line stores. The switching from  $45^\circ$  to  $135^\circ$  kernels and vice versa can preferably be done immediately. At the same time, the Interpolator 1231 searches for the maximum and minimum of the  $4 \times 4$  neighbourhood, and clamps the output pixel between them. The Interpolator 1231 can preferably produce one pixel per cycle, and is preferably synchronous to the main system clock

Fig. 24 shows the hardware architecture used for output clamping in accordance with the preferred embodiment of the present invention. The output clamping hardware of Fig. 24 fetches the  $(p-2)^{\text{th}}$ ,  $(p-1)^{\text{th}}$ ,  $p^{\text{th}}$  and  $(p+1)^{\text{th}}$  pixels from 4 line stores 1251 to 1255 in the ILSM 1207 and stores the pixels in one of two shift registers 2401 and 2403. The shift register 2401 is used to find the maximum value of the  $(p-2)^{\text{th}}$ ,  $(p-1)^{\text{th}}$ ,  $p^{\text{th}}$  and  $(p+1)^{\text{th}}$  pixels using a maximum finder 2405. The maximum value from the maximum finder

2405 is compared with the interpolated pixel value using a comparator 2407. The output of the comparator is fed to a multiplexer 2409 and if the calculated interpolated pixel value is greater than the maximum value, then the output value is replaced with the local maximum. Similarly, the minimum value of the  $(p-2)^{th}$ ,  $(p-1)^{th}$ ,  $p^{th}$  and  $(p+1)^{th}$  pixels from the 4 line stores 1251 to 1255 is found using the shift register 2403 and a minimum finder 2411. The minimum value is compared with the interpolated pixel value using the comparator 2413. The output of the comparator 2413 is fed to the multiplexer 2409 and if the calculated interpolated pixel value is less than the minimum value, then the output value is replaced with the local minimum.

The Colour Space Converter (CSC) 1233 converts YUV pixel data to RGB or vice versa. The equation for converting YUV from RGB is as follows:

$$\begin{pmatrix} Y \\ U \\ V \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.16875 & -0.33126 & 0.5 \\ 0.5 & -0.41869 & -0.08131 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

And the equation for converting RGB from YUV is as follows:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1.402 \\ 1 & -0.3441 & -0.7141 \\ 1 & 1.772 & 0 \end{pmatrix} \cdot \begin{pmatrix} Y \\ U \\ V \end{pmatrix}$$

The equations are implemented in hardware using nine multipliers (not illustrated) and nine adders (not illustrated) which add the products. The output of the matrix multiplication is clamped to 0 and 255. The CSC 1233 generates one pixel per cycle and is synchronous to the main system clock.

The Output Synchroniser 1235 synchronises the data from the main system clock domain to the output pixel clock domain. The main system clock rate should always be greater than or equal to the output pixel rate. The Output Synchroniser 1235 has a ping-pong buffer (not illustrated) which allows the pixels to be written into and read from the buffer at different rates. When one line store in the ping-pong buffer is full, the other line store is swapped to accept the incoming pixels in a next line, while the full line store is read at the output pixel clock rate. The Output Synchroniser 1235 receives an output pixel enable signal from the Output Signal Generator 1237. When the enable signal is asserted the Output Synchroniser 1235 should output a pixel to an external device, such as the display device 1014 of the computer system 1000. The Total storage required by the Output Synchroniser 1235 depends on the maximum width of the output display. If the maximum width is from HDTV, then total storage required is 2x2048x8bits, ie. 4kbytes.

The Serial ROM Interface (SRI) 1201 preferably follows the SPI™ protocol and has dual functions. When self configuration is selected the SRI 1201 assumes that a serial ROM is connected to it and will read all information required for the RC Chip 1103 operation. Otherwise, the SRI 1201 can be used as a means to communicate with another device, where the RC Chip 1103 acts as a target.

A '1394 Interface' 1263 is preferably used in the RC chip 1103. The '1394 Interface' 1263 is the IEEE designation for a new high-performance serial bus designed to fill the need for high-speed digital communications between electronic devices. The IEEE standard defines both a physical layer and a cable connected virtual bus. The interface standard defines a transmission method, media and protocol. The main advantage of the '1394 Interface' 1263 is firstly a large bandwidth capacity. The maximum 400Mbps bandwidth allows the '1394 Interface' 1263 to be used for streaming digital images. Using the '1394 Interface' 1263 the RC Chip 1103 can be connected to any digital still camera (and other multimedia devices) and can perform resolution conversion on images generated by the digital still camera. Secondly, "hot plugging," power sourcing, and dynamic reconfiguration abilities make the IEEE 1394 Interface a user-friendly interface. The features of the IEEE 1394 Interface will make plugging into a computer expansion system as easy as plugging into AC power, providing communications on demand without having to shut down and reconfigure the RC Chip 1103 each time an I/O device is added or removed. The RC Chip 1103 can be used as the image source rather than a Video Interface Controller.

The PCI Interface 1200 allows the host CPU, for example the processor 1005, to read and write various tables, line stores and registers in the RC chip 1103. The PCI Interface 1200 allows the host CPU to configure the RC Chip 1103 and perform diagnosis on the system when an error or interrupt occurs. The PCI Interface 1200 also allows the RC Chip 1103 to interrupt the host CPU under user specified circumstances. The PCI interface 1200 can be replaced by any microprocessor interface (eg., PowerPC™ interface) depending on the microprocessor that is used to configure the RC Chip 1103.

The foregoing describes only some embodiments of the present invention, and modifications and/or changes can be made thereto without departing from the scope and spirit of the invention, the embodiment(s) being illustrative and not restrictive.

**Claims:**

1. A method of clamping the output values of filtered image data comprising a mapping of discrete sample values, said method comprising the steps of:

5 for each discrete sample value of said mapping:

determining a maximum sample value and a minimum sample value of a plurality of input discrete samples values used to calculate said discrete sample value; and

clamping the output value of said discrete sample value to the domain of said plurality of input discrete sample values utilising said maximum sample value and said  
10 minimum sample value, wherein said output value is dependent on a plurality of attributes of said plurality of input discrete sample values.

2. The method according to claim 1, wherein said plurality of attributes includes a number representing the total number of colours represented by said plurality of input  
15 discrete sample values.

3. The method according to claim 2, wherein said number is compared to a threshold value.

20 4. The method according to claim 3, wherein said threshold is predetermined.

5. The method according to claim 3, wherein said threshold is dependent on said plurality of input discrete sample values.

25 6. The method according to claim 3, wherein said threshold value is 8.

7. The method according to claim 1, wherein said plurality of attributes includes a magnitude of said discrete sample value.

30 8. The method according to claim 7, wherein said magnitude is compared to said maximum sample value and said minimum sample value.

9. A method of interpolating image data comprising a plurality of discrete sample values, said method comprising the steps of:

accessing at least one portion of said discrete sample values of said image data;

calculating kernel values for each discrete sample value of said portion using one

5 of a plurality of kernels;

convolving said kernel values with said portion of discrete sample values to produce an output value; and

clamping said output value to the domain of said portion of discrete sample values, wherein said output value is dependent on a plurality of attributes of said portion  
10 of discrete sample values.

10. The method according to claim 9, wherein said plurality of attributes includes a number representing the total number of colours represented by said plurality of input discrete sample values.

11. The method according to claim 10, wherein said number is compared to a threshold value.

12. The method according to claim 11, wherein said threshold is predetermined.

13. The method according to claim 11, wherein said threshold is dependent on said plurality of input discrete sample values.

14. The method according to claim 11, wherein said threshold value is 8.

15. The method according to claim 9, wherein said plurality of attributes includes a magnitude of said discrete sample value.

16. The method according to claim 15, wherein said magnitude is compared to a maximum sample value and a minimum sample value of said portion of discrete sample values.

17. The method according to claim 16, wherein said output value is set to said maximum sample value if:



said number is less than said threshold value; and  
said output value is greater than said maximum sample value of said portion.

18. The method according to claim 8, wherein said output value is set to said minimum  
5 sample value if:

said number is less than said threshold value; and  
said output value is less than said minimum sample value of said portion.

19. The method according to claim 16, wherein said maximum sample value and said  
10 minimum sample value are calculated over all colour channels of said image data.

20. An apparatus for clamping the output values of filtered image data comprising a  
mapping of discrete sample values, said apparatus comprising:

- means for determining for each discrete sample value of said mapping, a maximum  
15 sample value and a minimum sample value of a plurality of input discrete samples values  
used to calculate said discrete sample value; and

- means for clamping the output value of said discrete sample value to the domain  
of said plurality of input discrete sample values utilising said maximum sample value and  
said minimum sample value, wherein said output value is dependent on a plurality of  
20 attributes of said plurality of input discrete sample values.

21. The apparatus according to claim 20, wherein said plurality of attributes includes a  
number representing the total number of colours represented by said plurality of input  
discrete sample values.

22. The apparatus according to claim 21, wherein said number is compared to a  
threshold value.

23. The apparatus according to claim 22, wherein said threshold is predetermined.

24. The apparatus according to claim 22, wherein said threshold is dependent on said  
plurality of input discrete sample values.

25. The apparatus according to claim 22, wherein said threshold value is 8.

26. The apparatus according to claim 20, wherein said plurality of attributes includes a magnitude of said discrete sample value.

5 27. The apparatus according to claim 26, wherein said magnitude is compared to said maximum sample value and said minimum sample value.

28. An apparatus for interpolating image data comprising a plurality of discrete sample values, said apparatus comprising:

10 access means for accessing at least one portion of said discrete sample values of said image data;

calculation means for calculating kernel values for each discrete sample value of said portion using one of a plurality of kernels;

15 convolution means for convolving said kernel values with said portion of discrete sample values to produce an output value; and

clamp means for clamping said output value to the domain of said portion of discrete sample values, wherein said output value is dependent on a plurality of attributes of said portion of discrete sample values.

20 29. The apparatus according to claim 28, wherein said plurality of attributes includes a number representing the total number of colours represented by said plurality of input discrete sample values.

30. The apparatus according to claim 29, wherein said number is compared to a threshold value.

31. The apparatus according to claim 30, wherein said threshold is predetermined.

32. The apparatus according to claim 30, wherein said threshold is dependent on said plurality of input discrete sample values.

33. The apparatus according to claim 30, wherein said threshold value is 8.

34. The apparatus according to claim 28, wherein said plurality of attributes includes a magnitude of said discrete sample value.

35. The apparatus according to claim 34, wherein said magnitude is compared to a maximum sample value and a minimum sample value of said portion of discrete sample values.

36. The apparatus according to claim 35, wherein said output value is set to said maximum sample value if:

said number is less than said threshold value; and

said output value is greater than said maximum sample value of said portion.

37. The apparatus according to claim 27, wherein said output value is set to said minimum sample value if:

said number is less than said threshold value; and

said output value is less than said minimum sample value of said portion.

38. The apparatus according to claim 35, wherein said maximum sample value and said minimum sample value are calculated over all colour channels of said image data.

39. A computer readable medium, having a program recorded thereon, where the program is configured to make a computer execute a procedure to clamp the output values of filtered image data comprising a mapping of discrete sample values, said program comprising:

code for determining for each discrete sample value of said mapping, a maximum sample value and a minimum sample value of a plurality of input discrete samples values used to calculate said discrete sample value; and

code for clamping the output value of said discrete sample value to the domain of said plurality of input discrete sample values utilising said maximum sample value and said minimum sample value, wherein said output value is dependent on a plurality of attributes of said plurality of input discrete sample values.

40. The computer readable medium according to claim 39, wherein said plurality of attributes includes a number representing the total number of colours represented by said plurality of input discrete sample values.

5 41. The computer readable medium according to claim 40, wherein said number is compared to a threshold value.

42. The computer readable medium according to claim 41, wherein said threshold is predetermined.

10

43. The computer readable medium according to claim 41, wherein said threshold is dependent on said plurality of input discrete sample values.

15

44. The computer readable medium according to claim 41, wherein said threshold value is 8.

45. The computer readable medium according to claim 39, wherein said plurality of attributes includes a magnitude of said discrete sample value.

20

46. The computer readable medium according to claim 45, wherein said magnitude is compared to said maximum sample value and said minimum sample value.

25

47. A computer readable medium, having a program recorded thereon, where the program is configured to make a computer execute a procedure to interpolate image data comprising a plurality of discrete sample values, said program comprising:

code for accessing at least one portion of said discrete sample values of said image data;

code for calculating kernel values for each discrete sample value of said portion using one of a plurality of kernels;

30

code for convolving said kernel values with said portion of discrete sample values to produce an output value; and

code for clamping said output value to the domain of said portion of discrete sample values, wherein said output value is dependent on a plurality of attributes of said portion of discrete sample values.

48. The computer readable medium according to claim 47, wherein said plurality of attributes includes a number representing the total number of colours represented by said plurality of input discrete sample values.

5

49. The computer readable medium according to claim 48, wherein said number is compared to a threshold value.

50. The computer readable medium according to claim 49, wherein said threshold is  
10 predetermined.

51. The computer readable medium according to claim 49, wherein said threshold is dependent on said plurality of input discrete sample values.

52. The computer readable medium according to claim 49, wherein said threshold value  
15 is 8.

53. The computer readable medium according to claim 47, wherein said plurality of attributes includes a magnitude of said discrete sample value.

20

54. The computer readable medium according to claim 53, wherein said magnitude is compared to a maximum sample value and a minimum sample value of said portion of discrete sample values.

25 55. The computer readable medium according to claim 54, wherein said output value is set to said maximum sample value if:

said number is less than said threshold value; and

said output value is greater than said maximum sample value of said portion.

30 56. The computer readable medium according to claim 46, wherein said output value is set to said minimum sample value if:

said number is less than said threshold value; and

said output value is less than said minimum sample value of said portion.

57. The computer readable medium according to claim 54, wherein said maximum sample value and said minimum sample value are calculated over all colour channels of said image data.

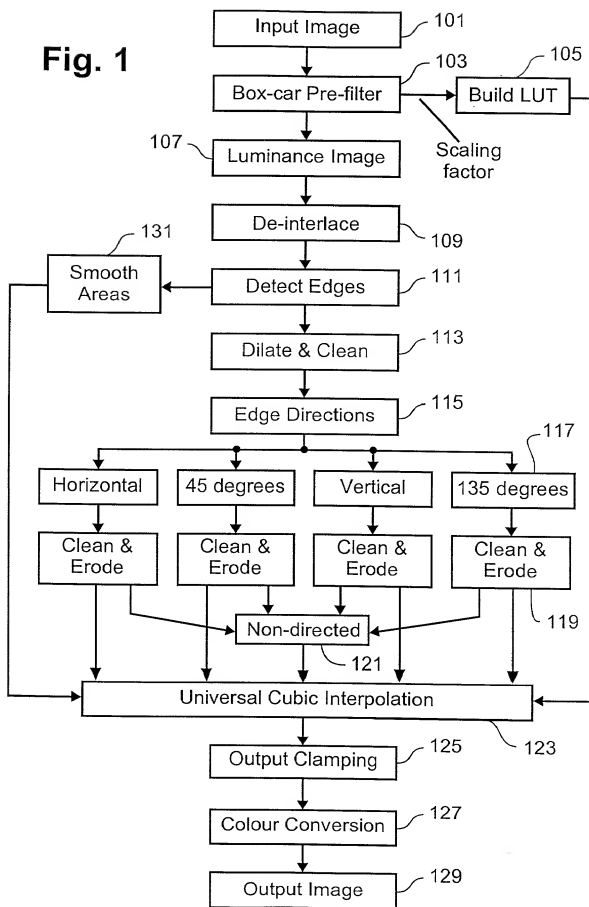
5

110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208  
2209  
2210  
2211  
2212  
2213  
2214  
2215  
2216  
2217  
2218  
2219  
2220  
2221  
2222  
2223  
2224  
2225  
2226  
2227  
2228  
2229  
2230  
2231  
2232  
2233  
2234  
2235  
2236  
2237  
2238  
2239  
2240  
2241  
2242  
2243  
2244  
2245  
2246  
2247  
2248  
2249  
2250  
2251  
2252  
2253  
2254  
2255  
2256  
2257  
2258  
2259  
2260  
2261  
2262  
2263  
2264  
2265  
2266  
2267  
2268  
2269  
2270  
2271  
2272  
2273  
2274  
2275  
2276  
2277  
2278  
2279  
2280  
2281  
2282  
2283  
2284  
2285  
2286  
2287  
2288  
2289  
2290  
2291  
2292  
2293  
2294

## Abstract

### COLOUR CLAMPING

- 5 Disclosed is a method and apparatus for clamping the output values of filtered image data comprising a mapping of discrete sample values are disclosed. For each discrete sample value of the mapping a maximum sample value and a minimum sample value of a plurality of input discrete samples values used to calculate the discrete sample value, is determined. The output value of the discrete sample value is clamped to the domain of
- 10 the plurality of input discrete sample values utilising the maximum sample value and the minimum sample value, wherein the output value is dependent on a plurality of attributes of the plurality of input discrete sample values.

**Fig. 1**



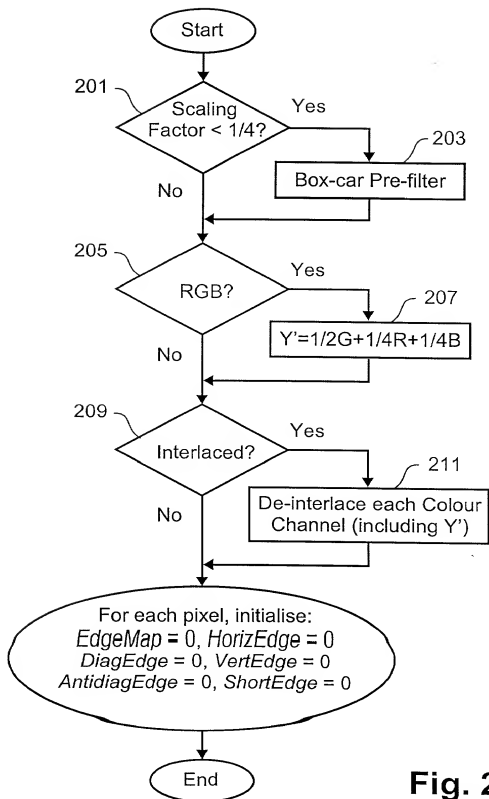


Fig. 2

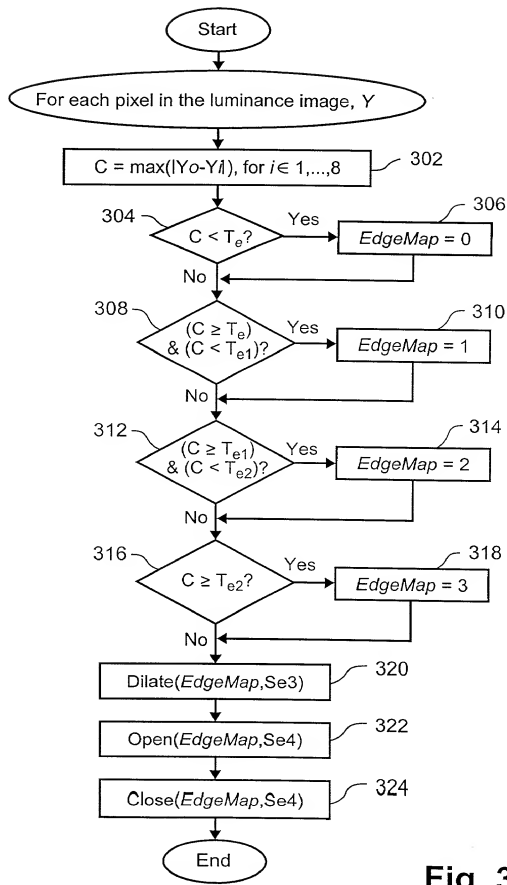


Fig. 3

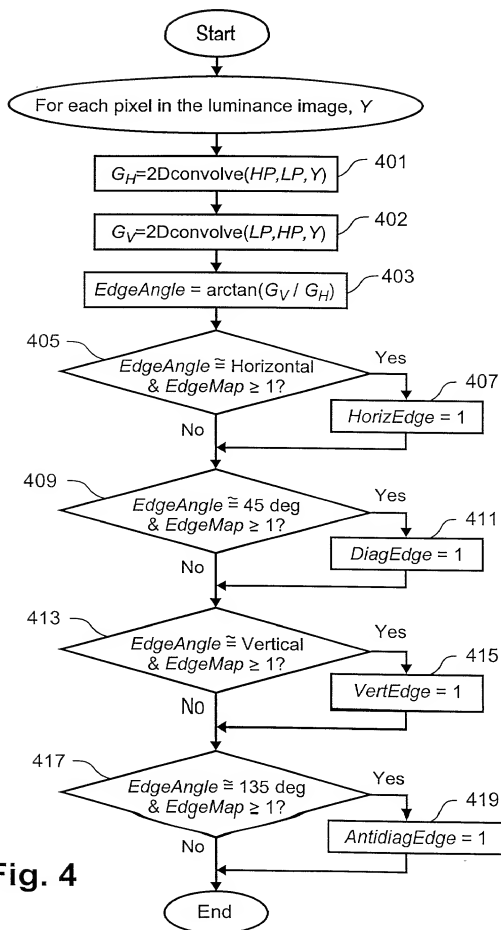
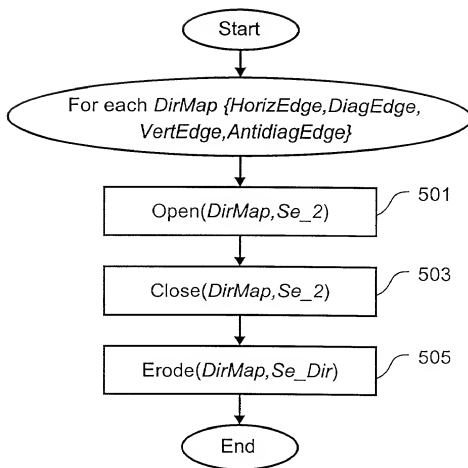
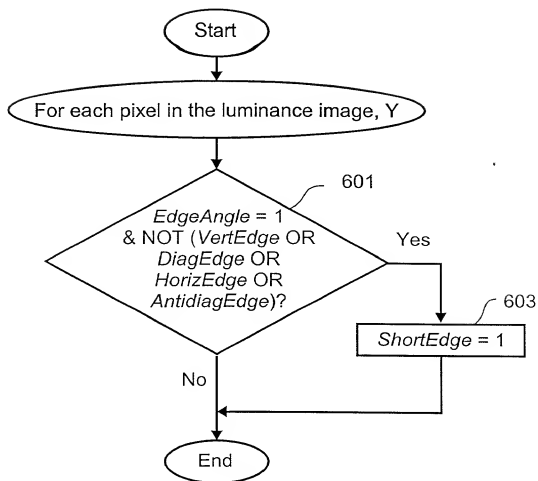


Fig. 4

**Fig. 5**

**Fig. 6**

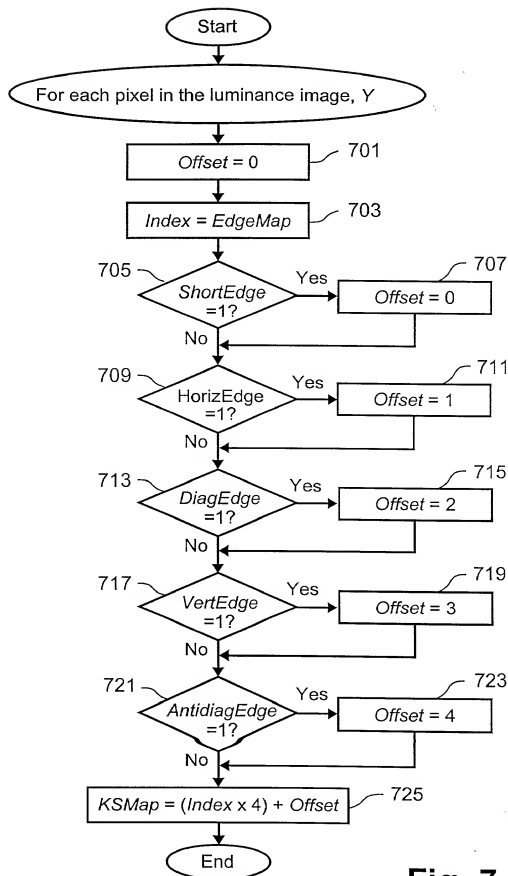
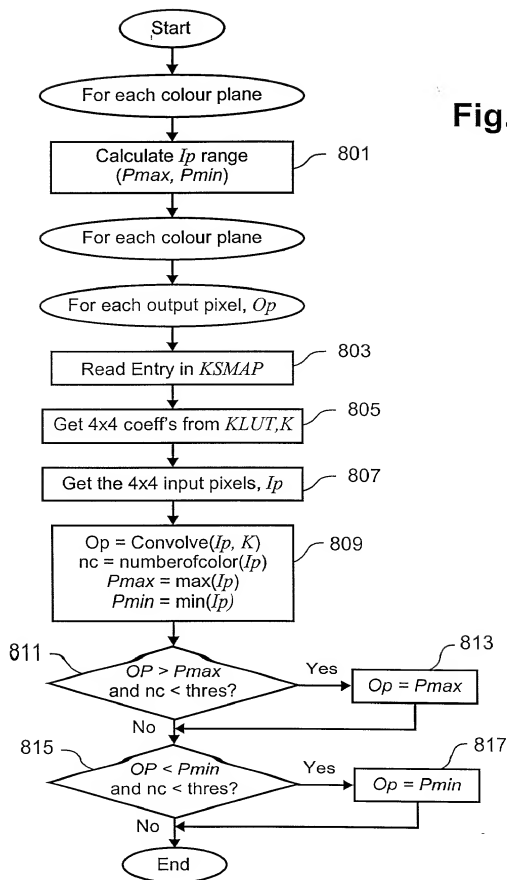


Fig. 7

Fig. 8



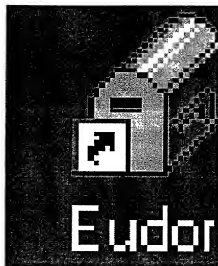


Fig. 9(a)



Fig. 9(b)



Fig. 9(c)



Fig. 9(d)



Fig. 9(e)



Fig. 9(f)





Fig. 9(g)



Fig. 9(h)

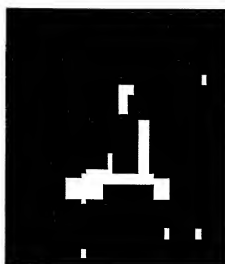


Fig. 9(i)



Fig. 9(j)

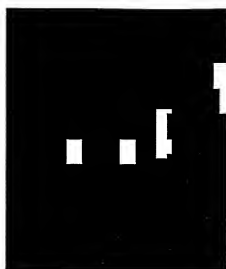


Fig. 9(k)

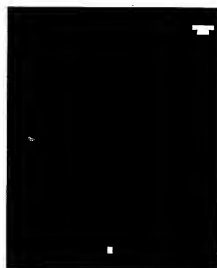


Fig. 9(l)



Fig. 9(m)

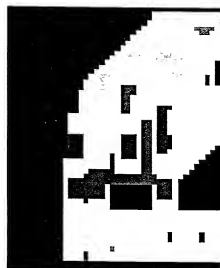
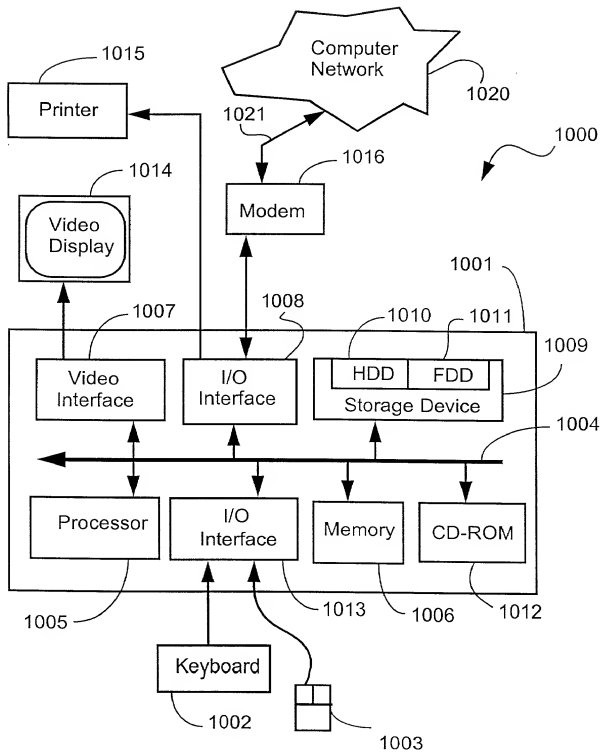
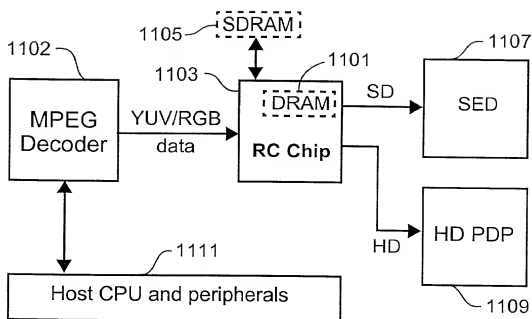


Fig. 9(n)

11/30

**Fig. 10**

**Fig. 11**

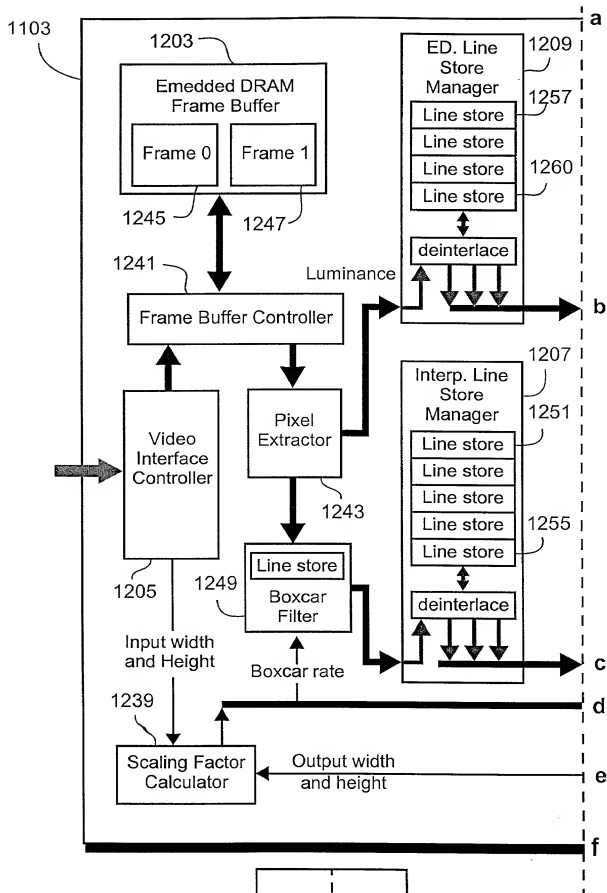
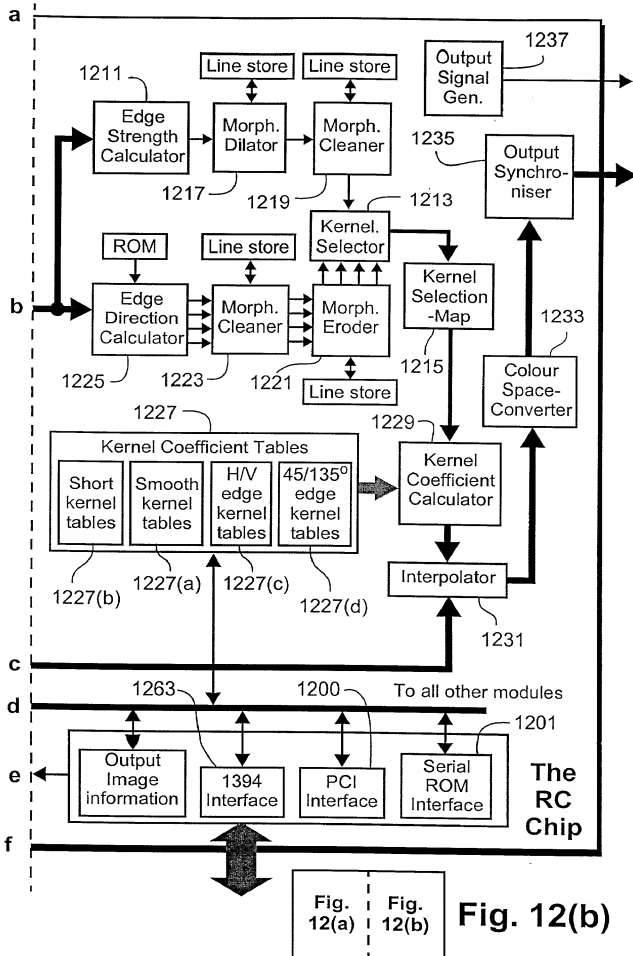


Fig. 12(a)

Fig. 12(b)



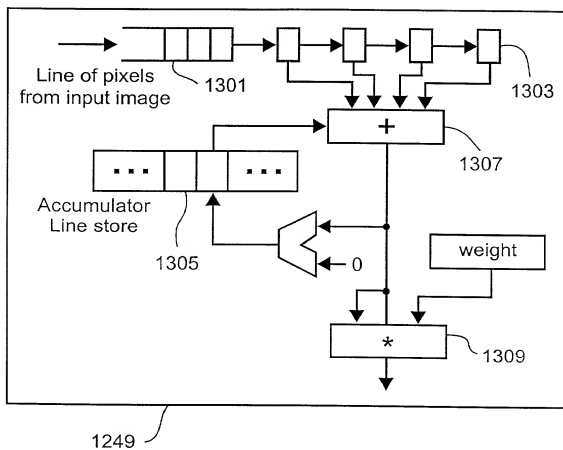


Fig. 13

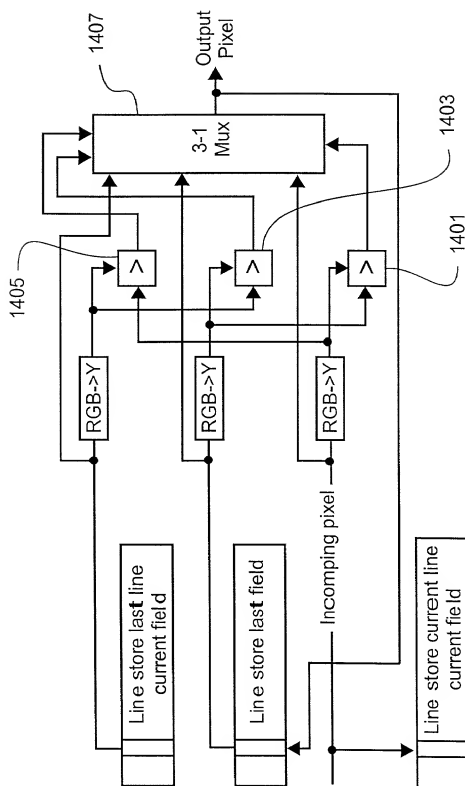
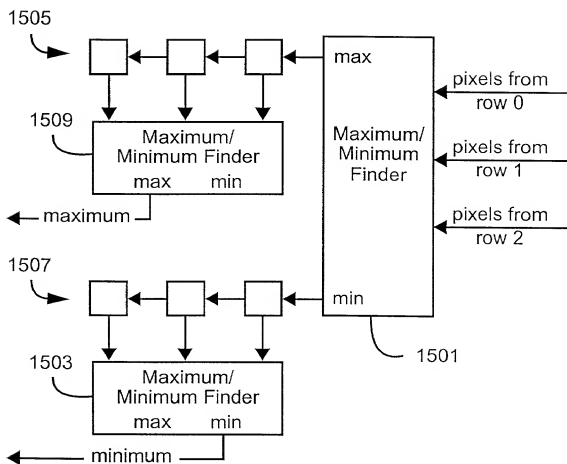
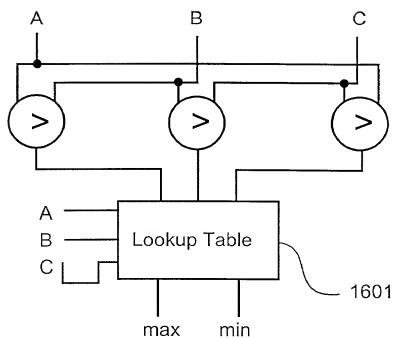


Fig. 14



**Fig. 15**

**Fig. 16**

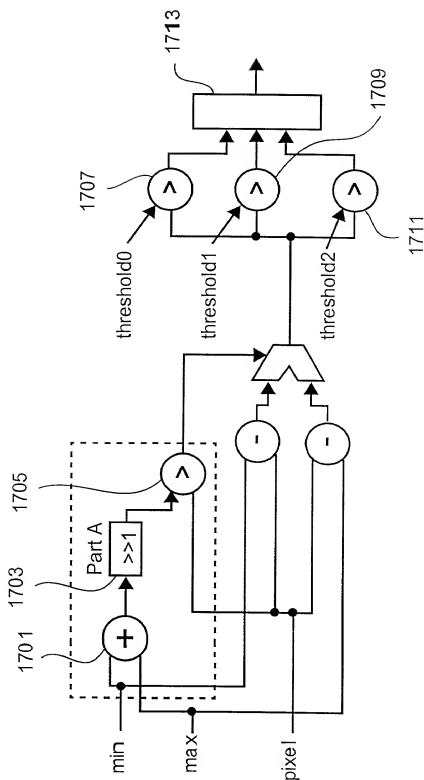
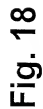


Fig. 17



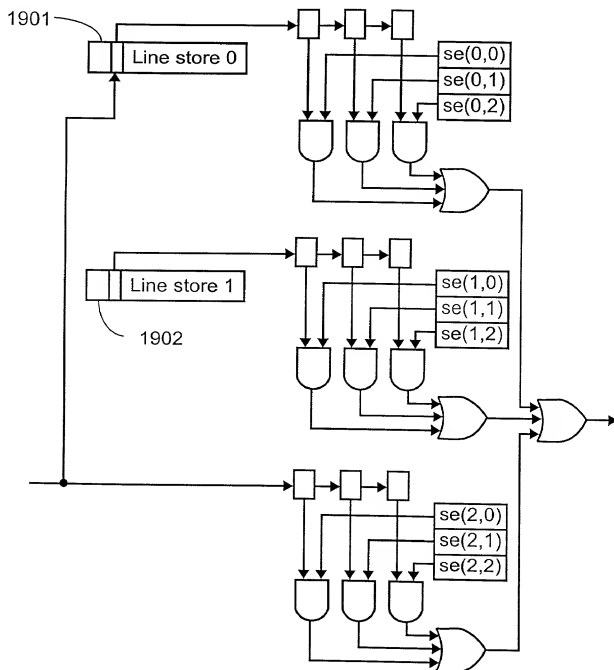


Fig. 19

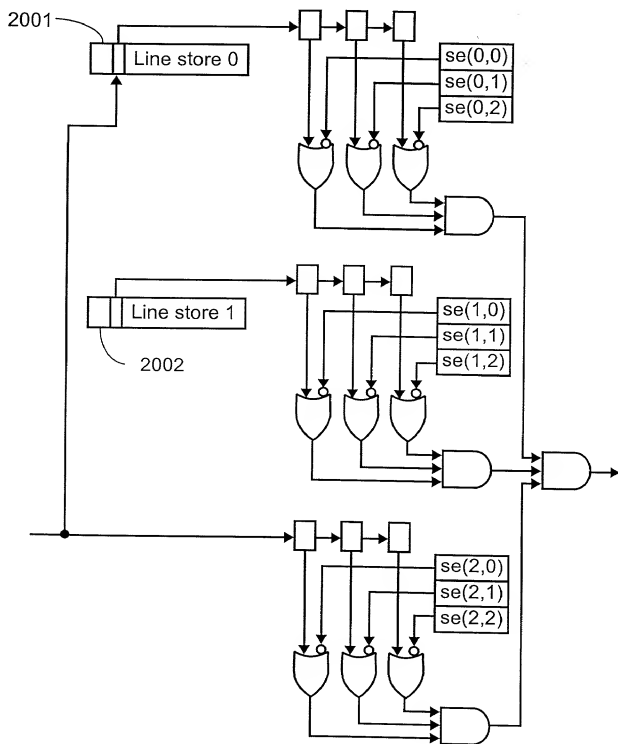
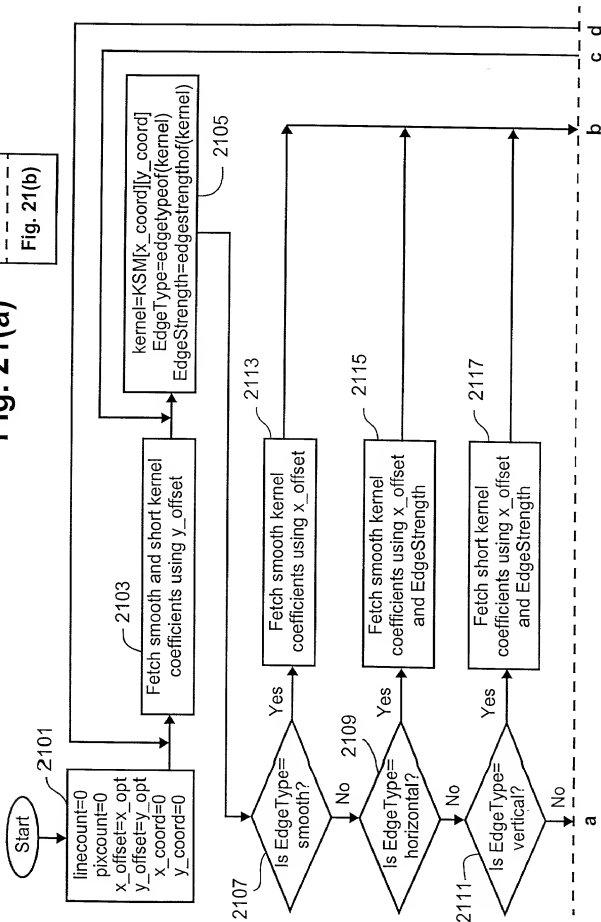


Fig. 20

Fig. 21(a)

Fig. 21(a)

Fig. 21(b)







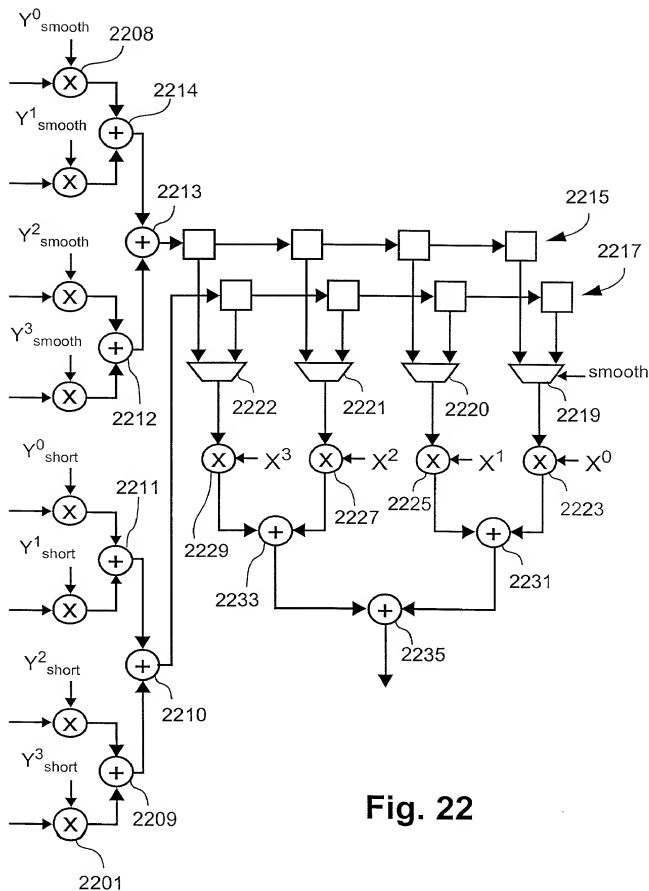
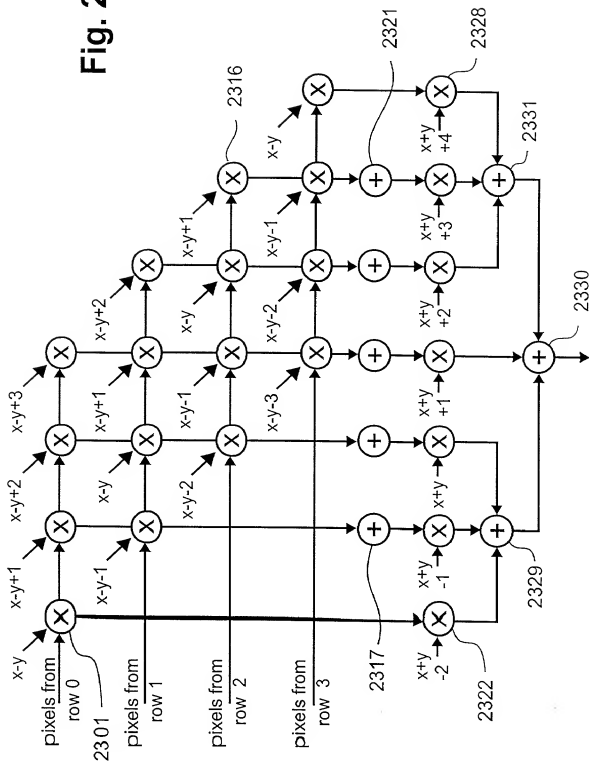


Fig. 22

Fig. 23



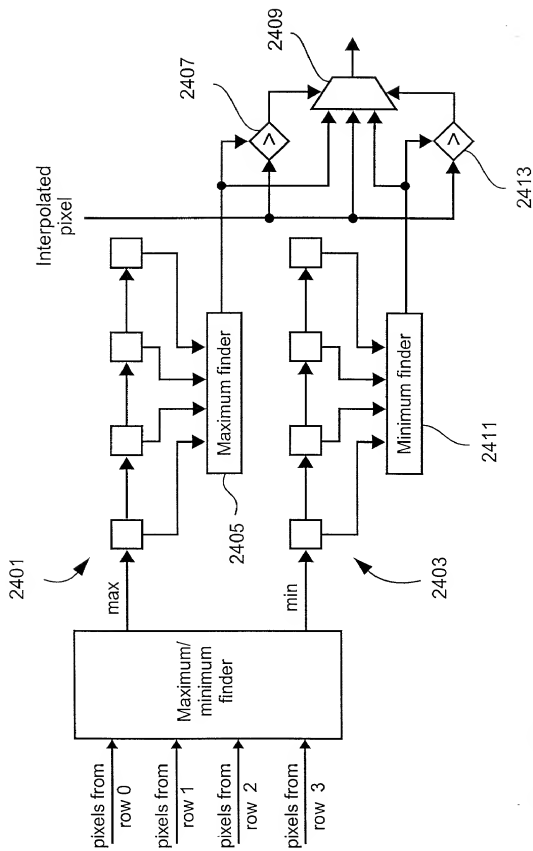
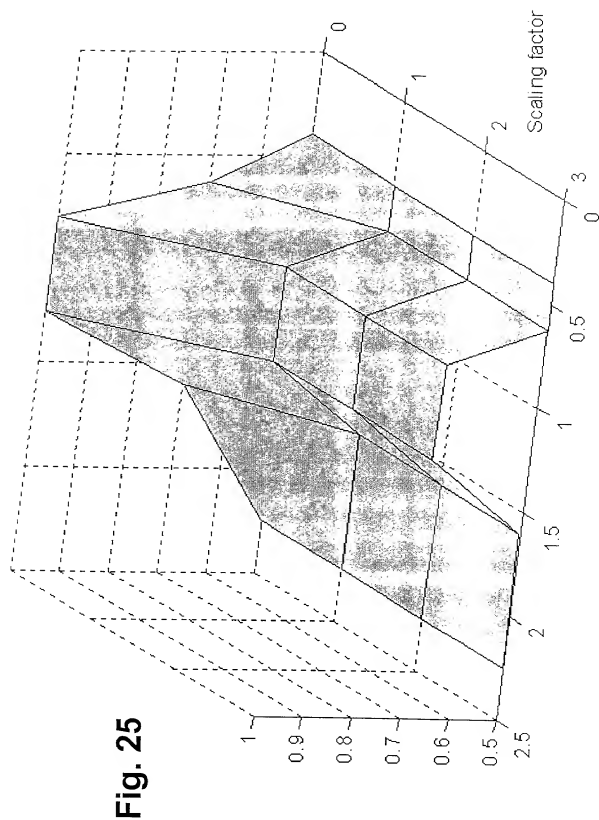


Fig. 24



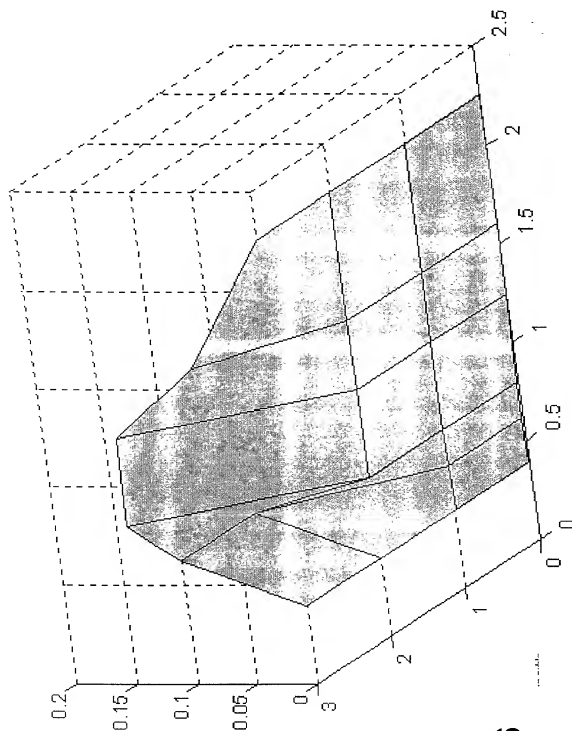


Fig. 26